Benchmarking full version of GureKDDCup, UNSW-NB15, and CIDDS-001 NIDS datasets using rolling-origin resampling

Article *in* Information Security Journal A Global Perspective · October 2021

DOI: 10.1080/19393555.2021.1985191

Yee Jian Chew
Multimedia University

Shih Yin Ooi
Multimedia University

Kok-Seng Wong
VinUniversity

Ying Han Pang
Multimedia University

Benchmarking full version of GureKDDCup, UNSW-NB15, and CIDDS-001 NIDS datasets using rolling-origin resampling

Yee Jian Chew, Nicholas Lee, Shih Yin Ooi, Kok-Seng Wong & Ying Han Pang

Benchmarking Full Version of GureKDDCup, UNSW-NB15, and CIDDS-001 NIDS Datasets using Rolling-origin Resampling

Yee Jian Chew[a]*, Nicholas Lee[a], Shih Yin Ooi[a], Kok-Seng Wong[b] and Ying Han Pang[a]

[a] *Faculty of Information Science and Technology, Multimedia University, Melaka, Malaysia;* [b] *College of Engineering and Computer Science, VinUniversity, Hanoi, Vietnam*

chewyeejian@gmail.com

"This is an Accepted Manuscript of an article published by Taylor & Francis in Information Security Journal: A Global Perspective on 19 Oct 2021, available at: https://www.tandfonline.com/doi/full/10.1080/19393555.2021.1985191"

Benchmarking Full Version of GureKDDCup, UNSW-NB15, and CIDDS-001 NIDS Datasets using Rolling-origin Resampling

Network intrusion detection system (NIDS) is a system that analyses network traffic to flag malicious traffic or suspicious activities. Several recent NIDS datasets have been published, however, the lack of baseline experimental results on the full version of datasets had made it difficult for researchers to perform benchmarking. As the train-test distribution of the datasets has yet to be pre-defined by the creators, this further obstruct the researchers to compare the performance unbiasedly across each of the machine classifiers. Moreover, cross-validation resampling scheme have also been addressed in the literatures to be inappropriate in the domain of NIDS. Thus, rolling-origin – a standard resampling technique which is also known as a common cross-validation scheme in the forecasting domain is employed to allocate the training and testing distributions. In this paper, rigorous experiments are conducted on the full version of the three recent NIDS datasets: GureKDDCup, UNSW-NB15, and CIDDS-001. While the datasets chosen might not be the latest available datasets, we have selected them as they include the essential IP addresses fields which are usually missing or removed due to some sort of privacy concerns. To deliver the baseline empirical results, 10 well-known classifiers from Weka are utilized.

Keywords: Network Intrusion Detection System (NIDS), Baseline, Benchmark, Sampling, Rolling-origin, Cross-Validation, Machine Classifier, GureKDDCup, UNSW-NB15, CIDDS-001

1 Introduction

As the number of coordinated cyberattacks are observed to be escalating swiftly over the past few years, numerous works have been developed by the research communities to enhance the capability of an intrusion detection system (IDS) for safeguarding an individual or organization against cyber threats (Mishra et al., 2018; Singh & Silakari, 2009). In particular, NIDS is designed to detect malicious traffic from network traces, and this can be achieved by creating a classification model to detect malicious traffic. With the explosive growth of computer vision, a number of machine learning techniques were predominantly proposed to segregate the malicious traffics from the benign traffic.

Although many machine algorithms have been enhanced to improve the detection rate of malicious traffic, most of the studies tend to perform their experiments on the benchmark datasets: KDDCup'99 (Stolfo et al., 2000) and NSL-KDD (Tavallaee et al., 2009). Several researchers (Ahmed et al., 2016; Mahoney & Chan, 2003; McHugh, 2000; Tavallaee et al., 2009) have critiqued the datasets as being outdated , since they were generated by the Defense Advanced Research Projects Office Agency (DARPA) for well over two decades ago.

Sommer and Paxson (2010) and Małowidzki et al. (2015) have pointed out that the lack of a recent representative publicly available NIDS datasets remains the biggest challenges in this domain. Following their comments, several NIDS datasets have been published, for instance, UNSW-NB15 (Moustafa & Slay, 2015) and CIDDS-001 (Ring et al., 2017) datasets have been made available online by the Australian Centre for Cyber Security and the University of Coburg.

In this paper, UNSW-NB15 (Moustafa & Slay, 2015) and CIDDS-001 (Ring et al., 2017) have been selected to be rigorously experimented as they contain modern attacks that are typically not observed in KDDCup'99. Both datasets mentioned are dedicated to tackle a different drawback by the previous benchmark KDDCup'99. Particularly, UNSW-NB15 complement for attacks which have low footprints by introducing 10 attack class (e.g., fuzzers, backdoors, exploits, reconnaissance, etc.) into the dataset (Moustafa & Slay, 2015), while CIDDS-001 contains a number of sophisticated attack scenarios associated with ping scanning, port scanning, brute force, and denial-of-service attacks (Ring et al., 2017). Additionally, GureKDDCup (Perona et al., 2016; Perona et al., 2008) is also chosen to be extensively experimented in this paper. Although GureKDDCup might not be the most recent NIDS datasets, considering it's methodology in mimicking the generation process of the benchmark KDDCup'99, GureKDDCup is able to maintain most of the features from KDDCup'99 while furnishing the missing payload information, IP addresses, and port numbers in KDDCup'99. All the three datasets that have been selected contain labelled instances that is necessary for training and evaluating a machine classifier. We refer to Ring et al. (2019) for a more comprehensive review for NIDS datasets.

Instead of experimenting on portion of the dataset, we utilized the full version of GureKDDCup, UNSW-NB15, and CIDDS-001. In general, there are two dominant reasons as to why the full version of these datasets are yet to be thoroughly examined. The primary reason is due to the enormous size of the datasets that would require a tremendous amount of computation resources that can lead to a very long processing time (Masduki & Ramli, 2016). Secondly, the full version of these datasets has not been partitioned into training and testing distribution by the original authors (Ring et al. 2019). This would imply that the researchers will not be able to fairly evaluate the performance of their classifiers alongside with other related work as each of us might have a distinct subset of train-test partitioned that tries to maximize the evaluation metrics (e.g., classification accuracy, detection rate, etc.). In practical, enormous volume of network traffics are generated every second, hence, classifiers adopted should be adequately competent to cope with the massive amount of network traffics. To prove the practicality of the machine classifiers, the full datasets are employed instead of the portion of the datasets. With the emerging trend of deep learning, huge amounts of data are necessary to be feed into the learning algorithms to build a high-performance algorithm (Ng, 2015). By conducting the experiments on the full version of datasets with machine classifiers, we have set forth a baseline performance for comparison purposes in the future.

Before evaluating the models, it is necessary to decide upon the type of sampling procedure to gauge the performance of machine algorithms. In general, a direct train-test evaluation scheme can be utilized if the datasets have been separated into a pre-defined split of training and testing distributions. However, training and testing sets are often not found in the full version of NIDS datasets, as they are simply not pre-distributed by the creators of the datasets. In the absence of train-test distribution, tenfold cross-validation (Kohavi, 1995) is an alternative approach which can be used to deliver unbiased and fair comparisons. While tenfold cross-validation have been recognized as a renowned evaluation scheme in multiple domains, such approach has deemed to be unsuitable in this domain, whereby the models will yield an over-realistic experimental results (Al Tobi & Duncan, 2019) and the possibility of leakage of test data into the training distribution (Ring et al., 2019). Thus, a distinct resampling scheme – rolling-origin, that is commonly use in the forecasting domain is adopted in this paper to avoid the biasness of cherry picking the training and testing distribution. Through some background studies, we noticed the lack of empirical experiments and results, specifically on the full version of the datasets. Hence, 10 distinct notable machine learning classifiers are utilized in this paper to empirically evaluate the aforementioned datasets.

The main contribution present in this paper are as follows: (i) the usage of rolling-origin resampling technique to partition the train-test distribution of the three NIDS datasets, (ii) the datasets cleaning and pre-processing procedure for the three NIDS datasets, and (iii) the utilization of 10 well-known classifiers to set forth a baseline empirical result. In Section 2, related experiments conducted on GureKDDCup, UNSW-NB15, and CIDDS-001 NIDS datasets are reviewed. As this section is meant to examine the concerns when adopting the datasets, the proposed technique in each of the literatures will not be deliberated extensively. The selection of the 10 machine learning classifiers and the rolling-origin evaluation approach is discussed in Section 3. Section 4 provides datasets cleansing procedure and the experimental results for the 10 machine algorithms on the three NIDS datasets. Lastly, Section 5 concludes the work.

2 Background Study

Most benchmarking papers that has been published aim to deliver a baseline performance predominantly for KDDCup'99 and NSL-KDD. Kayacık and Zincir-(Heywood 2005) employed K-means clustering and neural network algorithm on KDDCup-99 and two private NIDS datasets. In the work performed by Patil and Pattewar (2014), they compare the performance of AdaBoost against seven machine classifiers on KDDCup-99 and NSL-KDD. Divekar et al. (2018) have selected six classifiers to provide the benchmark experimental results on KDDCup'99, NSL-KDD, and UNSW-NB15. In their experimental studies, they utilized feature selection technique such as SMOTE oversampling (Chawla et al., 2002) and random undersampling on the datasets before delivering it to the six classifiers. It should be noted that the authors use the smaller version of UNSW-NB15 instead of the full version. Al Tobi and Duncan (2019) employed support vector machines, random forest and decision tree with threshold adaption and SMOTE feature selection to improve the detection rate on GureKDDCup, STA2018 (generated from ISCX dataset) (Shiravi et al., 2012) and two other synthetic datasets. Although the full version of GureKDDCup is employed, the evaluation strategy and sampling technique are slightly different from the work we proposed in this paper. For both of the studies (Al Tobi & Duncan, 2019; Divekar et al., 2018), feature selection is performed before evaluating the machine classifiers.

In general, the feature selection step is important to select the salient attributes and filter the unneeded, irrelevant and redundant attributes from the dataset to reduce the running time of a learning algorithm (Dash & Liu, 1997), improve the prediction performance (Guyon & Elisseeff, 2003), and simplify the models for easier interpretation (Bermingham et al., 2015). In this work, we do perform a simple filtering process where all redundant attributes were removed. However, we do not employ any feature selection techniques to automatically pick the "appeared to be important" attributes due to several reasons. First, we are not the domain expert to judge if the selected attributes do consequential in separating benign and malicious traffics. Furthermore, removing attributes will be harmful if all candidate attributes are equally relevant. Secondly, this step is omitted to avoid biasness towards certain classifiers so that the performances of each classifiers can be fairly judged (i.e., it is interesting to observe how a tree-based strategy can retrieve its own subset through the pruning process, etc.). It is also intentionally left out to avoid the feature subset selection bias because the performances of classifiers are evaluated using rolling-origin resampling method in this work. Thus, feature selection techniques are not utilized in this paper although it was employed in the previous studies conducted by Al Tobi & Duncan (2019) and Divekar et al. (2018).

While the work in the previous studies were interesting (Al Tobi & Duncan, 2019; Divekar et al., 2018), they do not include any exhaustive review on the full version of the GureKDDCup, UNSW-NB15, and CIDDS-001. Hence, detailed of past experiments performed on the three NIDS datasets are comprehensively explored.

2.1 Recent Dataset

For decades, KDDCup'99 (Stolfo et al., 2000) and NSL-KDD (Tavallaee et al., 2009) have been widely known as the benchmark datasets for the domain of NIDS. Considering their availability, many researchers have performed various classification task on the datasets (Ahmed et al., 2016). As the datasets were created way back in 1999, several literatures have criticized the datasets to be obsolete because it does not represent modern attack in the present (Ahmed et al., 2016; Mahoney & Chan, 2003; McHugh, 2000; Tavallaee et al., 2009). To resolve the issues raised, several NIDS datasets have been published recently. For instance, GureKDDCup (Perona et al., 2016; Perona et al., 2008), UNSW-NB15 (Moustafa & Slay, 2015) and CIDDS-001 (Ring et al., 2017) datasets have been made publicly available in conducting intrusion detection relevant tasks. Though some time have passed since the datasets have been made to be accessible publicly, only limited studies and experiments have been carried out on the aforementioned datasets. Particularly, the full version of the datasets has yet to be explored extensively due to the enormous amount of data, leading it to the computational resource dilemma (Elhag et al., 2017; Masduki & Ramli, 2016). In this section, discussions are focused predominantly on the experiments that have been conducted on GureKDDCup, UNSW-NB15, and CIDDS-001 datasets. As the primary intention of this section is to investigate the train-test data distribution and the approach undertaken to evaluate the models using the datasets, the proposed technique and classification accuracy in each of the literatures will not be deliberated in detailed since they are not the main concern in this study.

2.1.1 GureKDDCup

In 2008, GureKDDCup (Perona et al., 2016; Perona et al., 2008) dataset was generated according to the same process as KDDCup'99 but it additionally includes each pair of IP addresses, pair of port numbers, and a few new attacks that are not found in KDDCup'99. Apart from the full version GureKDDCup dataset, the creators of GureKDDCup also released a smaller version (6 percent) extracted from the entire GureKDDCup dataset.

While GureKDDCup was published more than 10 years ago, GureKDDCup dataset contains a much richer and cleaner features describing each connections in comparison towards KDDCup'99 (Stolfo et al., 2000). Additionally, creators of the datasets prepared the data in a way that it is suitable to be used for machine classification tasks further motivates us to investigate the datasets with various machine classifiers.

Table 1 summarizes all the experiments performed on the GureKDDCup dataset. Referring to the table, it can be observed that most of the researchers employed the 6 percent GureKDDCup version in their experimental settings except Al Tobi and Duncan (2019). Besides, it can also be seen that there are no standard schemes for allocating the percentage of train-test data and evaluating the models.

Al Tobi and Duncan (2019) adopted the prospective sampling method (File-to-File) in evaluating C5.0, random forest, and SVM. For example, the model is trained on File 1 and tested on File 2, 3 and 4; or being trained on File 2 and evaluated on File 1, 3 and 4. Although their work utilizes the full version of the dataset, the sampling and evaluation methods employed are completely different than the methodology proposed in this paper.

Table 1: Summary of the Experiments Conducted on GureKDDCup

Author(s)	Technique	Dataset(s)	#Train	#Test	#Class	Acc. (%)
Sahu and Jena (2014)	K-means Clustering	6% GureKDDCup	160,904	n/a	3	83.9
Abas et al. (2015)	Feature Selection + r-chunk Artificial Neural Network (ANN)	6% GureKDDCup	**n/a**	**500**	2	n/a
Sahu and Jena (2016)	Multi-Class Support Vector Machine Classifier (MSVM)	GureKDDCup	160,904	178,810	28	99.146
Ikram and Cherukuri (2016)	PCA + SVM with Automated Parameter Selection	10% GureKDDCup	n/a	10 CV	28	DR = 0.999
Masduki and Ramli (2016)	Feature Selection + SVM for R2L and DoS class	10% GureKDDCup	**90%**	**10%**	2	DR = 99.9735 (DoS) DR = 99.0297 (R2L)
Zhu et al. (2017)	Feature Selection with I-NGSA-III	6% GureKDDCup	n/a	10 CV	5	DR = 99.62 ± 0.17
Elhag et al. (2017)	Fuzzy Association Rule Mining with Multi-Objective Evolutionary Algorithm (FARC-HD with NGSA-II)	6% GureKDDCup	**10% (17,911)**	**90% (160,862)**	5	78.8
Jabbar et al. (2017)	K-mean + ADTree with K-Nearest Neighbour (KNN)	6% GureKDDCup	n/a	10 CV	2	99.93
Wang et al. (2017)	Logarithm Marginal Density Ratios Transformation-Support Vector Machines (LMDRT-SVM)	6% GureKDDCup	n/a	10 CV	2	99.18

Author(s)	Technique	Dataset(s)	#Train	#Test	#Class	Acc. (%)
Sainis et al. (2018)	Feature Selection + Outlier Removal with Interquartile Range	6% GureKDDCup	n/a	10 CV	n/a	99.08
Al-Riyami et al. (2018)	Long Short-Term Memory (LSTM)	6% GureKDDCup	**80%**	**20%**	n/a	F1 = 99.42
Al Tobi and Duncan (2019)	Batch Learning with Prediction threshold (cut-off)	Full GureKDDCup	n/a	10 CV	2	G-Acc. = 0.9998
			n/a	**File-to-File**	2	G-Acc. = 0.9998

DR – Detection Rate; CV – Cross-Validation;
G-Acc. – G-mean Accuracy; F1 – F1-Score
Not available (n/a) – not mentioned by author(s)

2.1.2 UNSW-NB15

Cyber Range Lab of the Australian Center for Cyber Security (ACCS) released the UNSW-NB15 dataset in 2015 to complement the lack of modern network traffic (Moustafa & Slay, 2015). Based on the original documentation, the full version of the dataset encompassed of 2,540,044. Due to the huge amount of instances, the authors also prepared a smaller version of the dataset containing predefined split of train-test distribution with the redundant records being discarded in both the training and testing sets (Moustafa & Slay, 2016). It should be noted that the smaller version does not include 5 features (scrip, sport, dstip, stime and ltime) that are found in the full version (Al-Zewairi et al., 2017; Janarthanan & Zargari, 2017). Additionally, Al-Zewairi et al. (2017) have verified that the full version of UNSW-NB15 contains three extra instances, leading to a total of 2,540,047 instances.

All of the research experiments conducted on the UNSW-NB15 are tabulated in Table 2. Aside from Al-Zewairi et al. (2017) which adopted the full version of the dataset to evaluate their model, other literatures employed the smaller version of the dataset that have been partitioned by the creators. Similar to the examination in Section 2.1.1 on

GureKDDCup, various train-test distributions and evaluation methods can also be observed from Table 2.

Table 2: Summary of the Experiments Conducted on UNSW-NB15

Author(s)	Technique	Dataset(s)	#Train	#Test	#Class	Acc. (%)
Guha et al. (2016)	Feature Selection with Genetic Algorithm + Artificial Neural Network	Part of UNSW-NB15	n/a	**119,747**	9	95.46
Gharaee and Hosseinvand (2016)	Feature Selection with Genetic Algorithm + Support Vector Machine (GF-SVM)	UNSW-NB15	n/a	n/a	10	91.22 (DoS)
Moustafa and Slay (2017)	Feature Selection + Association Rule Mining + Logistic Regression	Part of UNSW-NB15	175,341	82,332	2	83.0
Timčenko and Gajin, (2017)	Bagged Tree	UNSW-NB15	**40,000**	5 CV	10	DR = 92.6 (DoS)
Baig et al. (2017)	Cascade of Boosting-based Artificial Neural Network (CANID)	Part of UNSW-NB15	**82,232**	**175,341**	2	86.4
Al-Zewairi, Almajali, and Awajan (2017)	Multilayer Feedforward Artificial Neural Network	**Full UNSW-NB15**	**60%** (Train) **10%** (Validation)	**30%**	2	98.99
Belouch et al. (2017)	Feature Selection + REPTree	Part of UNSW-NB15	175,341	82,232	2	88.95
Benmessahel et al. (2017)	Multiverse Optimiser + Artificial Neural Network	Part of UNSW-NB15	175,341	82,232	2	99.61
Idhammad et al. (2017)	Artificial Neural Network-based DoS Detection Method (ADDM)	Part of UNSW-NB15 (109370 instances)	**60%**	**40%**	2	97.1

Author(s)	Technique	Dataset(s)	#Train	#Test	#Class	Acc. (%)
Primartha and Tama (2017)	Random Forest with 800 Trees	Part of UNSW-NB15	175,341	**10 CV**	2	95.5
Janarthanan and Zargari (2017)	Feature Selection + Random Forest	Part of UNSW-NB15	**82,332**	**175,341**	10	81.6175
Zhou et al. (2018)	Deep Feature Embedding Learning (DFEL) + Naïve Bayes	**20% of Part UNSW-NB15**	**70%**	**30%**	10	92.52
Idhammad et al. (2018c)	Online Sequential Semi-Supervised Machine Learning	Part of UNSW-NB15 (277705 instances)	**60%**	**40%**	2	93.71
Moustafa et al. (2018)	Beta Mixture Model-Anomaly-based IDS (BMM-ADS)	Part of UNSW-NB15	selected 50,000 to 200,000	82,232	2	93.4
Anwer et al. (2018)	Feature Selection Framework with Filter and Wrapper using J48 and Naïve Bayes	Part of UNSW-NB15	175,341	82,232	2	88.3

DR – Detection Rate; CV – Cross-Validation;
Not available (n/a) – not mentioned by author(s)
Part of UNSW-NB15 – train-test split by original author (Moustafa and Slay 2016)

2.1.3 CIDDS-001

CIDDS-001 (Coburg Intrusion Detection Dataset) dataset was created by emulating a small business environment encompassed of internal servers with OpenStack environment and external servers (Ring et al., 2017). To generate a realistic network traffic, user activities and behaviors are simulated with scripts by taking consideration of the working hours and styles. Besides, several distinct types of attacks have also been initiated and labelled by the authors. In order to capture up-to-date real traffic, the external servers are set up on the Internet, thereby, allowing the servers to be accessed publicly.

Table 3 compiles most of the recent experiments conducted on the CIDDS-001 dataset. As the dataset have not been partitioned into training and testing data by the authors, most of the studies are performed with different distribution and validation methods. Akin to the observations in GureKDDCup and UNSW-NB15 datasets, the allocation of train-test data and the validation strategy also vary from literature to literature.

Table 3: Summary of the Experiments Conducted on CIDDS-001

Author(s)	Technique	Dataset(s)	#Train	#Test	#Class	Acc. (%)
Elmasry et al. (2019)	Particle Swarm Optimisation + Deep Belief Network	5% of CIDDS-001	**800,000**	**800,000**	5	94.66
Tama and Rhee (2017)	Deep Neural Network	CIDDS-001 (146500 instances)	146,500	10 CV	2	99.99
			146,500	5 x 2CV	2	99.99
			70%	**30%**	2	99.99
Idhammad et al. (2018b)	Naïve Bayes Anomaly Detection + Random Forest in Cloud Environment	OpenStack CIDDS-001 (Week 1) (8451520 Instances)	**60%**	**40%**	4	97
Althubiti et al. (2018)	LSTM with rmsprop optimizer	External Server CIDDS-001	**67%** **(449,731)**	**33%** **(221,510)**	5	84.83
Idhammad et al. (2018a)	Statistical Network Entropy Anomaly Detection + Random Forest Classification for DDoS Attacks	CIDDS-001 (1st Day of Week 1) (1501857 Instances)	**60%**	**40%**	2	99.54%

Author(s)	Technique	Dataset(s)	#Train	#Test	#Class	Acc. (%)
Verma and Ranga (2018b)	K-nearest Neighbour	OpenStack CIDDS-001 (Week 1) (172839 Instances)	**66%**	**34%**	3	100
		External Server CIDDS-001 (Week 3) (153026 Instances)	**66%**	**34%**	5	99.6
Ring et al. (2018)	Unsupervised and Supervised Learning for Port Scanning	OpenStack CIDDS-001	Week 1	Week 2	2	n/a
Verma and Ranga (2018a)	Random Forest	OpenStack CIDDS-001	172,839	**Train Data**	3	100
		External Server CIDDS-001	153,026	**Train Data**	5	99.9
Nicholas et al. (2018)	LSTM	OpenStack CIDDS-001	Week 1 (8,415,1520)	Week 2 (10,310,733)	2	DR = 99.7896
Chen and Tsai (2018)	Search Economics with k-means clustering and support vector machine (SEKS)	CIDDS-001	**7,999**	**17,669**	5	93.28

DR – Detection Rate; CV – Cross-Validation;
Not available (n/a) – not mentioned by author(s)

2.2 NIDS Dataset Information

To further substantiate the observation in Section 2.1, the total number of instances contained in each of the aforementioned NIDS datasets are summarized in Table 4. From the table, it can be clearly seen that most of the datasets do not include a predefined split of training and testing data except 10% KDDCup'99, NSL-KDD, and part of UNSW-NB15. It should be noted that the filename for 'part of UNSW-NB15' dataset have been named incorrectly, whereby the train file and test file is named inversely.

Table 4: Summary of Datasets Information

Dataset(s)	**Filename**	**Total Instances**	**#Train**	**#Test**
KDDCup'99	`KDDCup99_full.arff`	4,898,430		
10% KDDCup'99	`KDDCup99.arff` [Train] KDDCUp_corrected_testing data (filename: `corrected` [Test])	805,049	494,020	311,029
NSL-KDD	`KDDTrain+.csv` [Train] `KDDTest+.csv` [Test]	148,516	125,973	22,543
6% GureKDDCup	`gureKddcup6percent.arff`	178,810		
Full GureKDDCup	Multiple File of `gureKddcup-matched.list`	2,759,494		

Part of UNSW-NB15
(Train-test file named inversely)
	`UNSW_NB15_testing-set.csv`
	[Train]
	`UNSW_NB15_training-set.csv`
	[Test]
		257,573	175,341	82,232

Full UNSW-NB15	`All UNSW-NB15`	2,540,047		

Dataset(s)	Filename	Total Instances	#Train	#Test
	UNSW-NB15_1.csv	700,001		
	UNSW-NB15_2.csv	700,001		
	UNSW-NB15_3.csv	700,001		
	UNSW-NB15_4.csv	440,044		
OpenStack CIDDS-001	All OpenStack CIDDS-001	31,287,933		
	CIDDS-001-internal-week1.csv	8,451,520		
	CIDDS-001-internal-week2.csv	10,310,733		
	CIDDS-001-internal-week3.csv	6,349,783		
	CIDDS-001-internal-week4.csv	6,175,897		
External Server CIDDS-001	All External Server CIDDS-001	23,009,251		
	CIDDS-001-external-week1.csv	172,838		
	CIDDS-001-external-week2.csv	10,310,733		
	CIDDS-001-external-week3.csv	6,349,783		
	CIDDS-001-external-week4.csv	6,175,897		

2.3 Discussions on the NIDS Datasets

As accentuated by Bamakan (2017), direct comparison of experimental results are not practical in most cases because it is an exhaustive task whereby consideration should be given on the pre-processing steps, experimental settings, sampling methods, evaluation metrics and etc. It is not even reasonable to directly compare the empirical results when the training and testing distribution is entirely different. As shown in Table 1 – Table 3, most of the experiments adopted a distinct set of distribution for building and evaluating the models. Additionally, the number of instances engage in some of the experiments are relatively small when compared against the total instances, as deliberated in Table 4.

Masduki and Ramli (2016) pointed out that it is computationally expensive to evaluate the models on the full version of NIDS datasets as it required very long processing time due to the massive number of instances found in each of the datasets. Besides, Elhag et al. (2017) mentioned that the scalability of current models might not be able to handle such enormous amount of data. Due to these reasons, limited studies have been performed using the full version of GureKDDCup, UNSW-NB15 and CIDDS-001. Hence, the full version of these datasets will be extensively experimented in this paper to complement the lack of experimental results.

While tenfold cross-validation (Kohavi, 1995) have been proven as one of the prominent evaluation techniques to provide fair results for comparison purposes, Ring et al. (2019) described that such approach might not be feasible in the case of intrusion detection. They justified that tenfold cross-validation is not suitable to be adopted in the domain of NIDS because there is a possibility that some of the testing data (e.g., flow of port scan) might be found in the training data during the splitting process of cross-validation. By using CIDDS-002 dataset as an example, they recommended to build the model by employing week one data while the data from week two is utilized in evaluating the model (and vice versa). On the other hand, the work by Al Tobi and Duncan (2019) revealed that the over-optimistic empirical results obtained from using cross-validation would not represent the genuine performance of the detection models in realistic scenarios. Thus, a distinct approach is utilized in this paper and the approach will be described thoroughly in Section 3.

3 Methodology

3.1 Rolling-origin Evaluation

By considering the latent interactivities and relationships between the network traffic, an amass of data instances are considered to be sequential, and they often exhibit some form of temporal properties (Bergmeir & Benítez, 2012). On this account, the standard and favored tenfold cross-validation (Kohavi, 1995) would not appropriate to be adopted in such setting, as it would undermines the temporal dependencies of the instances. In a typical forecasting task, training data should not contain future observations preceding the time step of the testing instances. Likewise, any observation prior to the timelines of the training set should not be included in the respective testing set (Hyndsight, 2016). Considering the domain of NIDS, the unsuitability of tenfold cross-validation is further substantiated by several literatures in Section 2.3.

As shown in Table 4, the train-test distribution of the selected datasets has not been predefined by the author(s). Thus, rolling-origin (Tashman, 2000) is employed in this paper, which is commonly adopted in forecasting tasks as an alternate cross-validation technique (Bergmeir & Benítez, 2012). Instead of adopting the standard practice to use a single distribution (e.g., a day) as the test set, all remaining data not utilized in training the model are employed. This seems to be a better approach as the majority of the forecasting tasks (e.g., stock market prediction) focuses on anticipating certain outcome (e.g., stock value) on a particular day, week or month in the future, as depicted in Figure 1. However, in the case of network intrusion detection, the data instances are not constrained to only a single point in time, but are unfolded indefinitely. Hence, the original rolling-origin evaluation is improvised as such: At any given fold, data instances (in the later time steps) that are yet to be used in training the model, are all adopted as the testing instances for the particular fold as shown in Figure 2. By using the improvised rolling-origin approach, both the training and testing data are distributed accordingly to the number of weeks available in the datasets (as shown in Table 7, Table 10 and Table 13). It is also worth mentioning that, for each of the train-test distributions, the machine model is required to be recalibrated and rebuilt in order to deliver the empirical results.

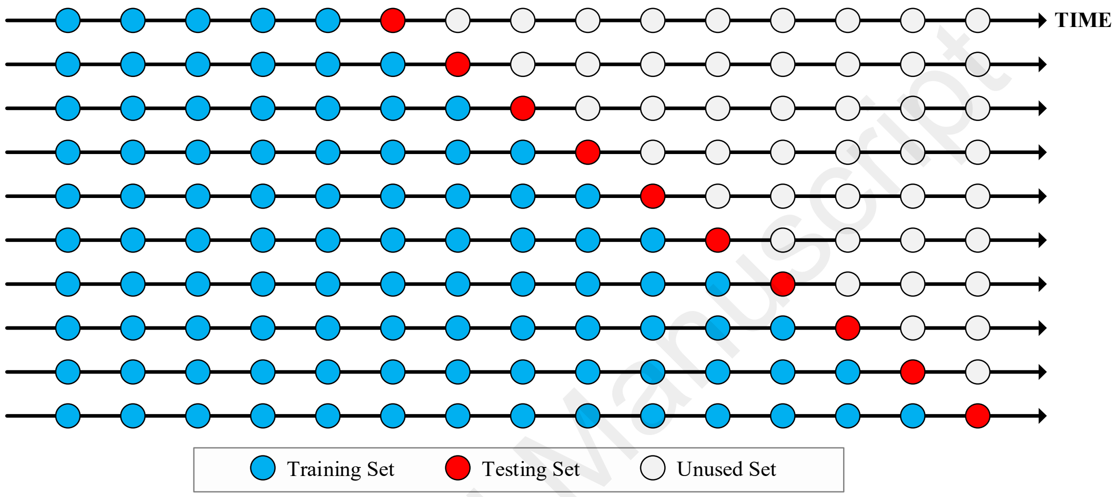


Figure 1: 1-step-ahead Rolling-origin Evaluation (Redrawn from (Hyndsight, 2016))

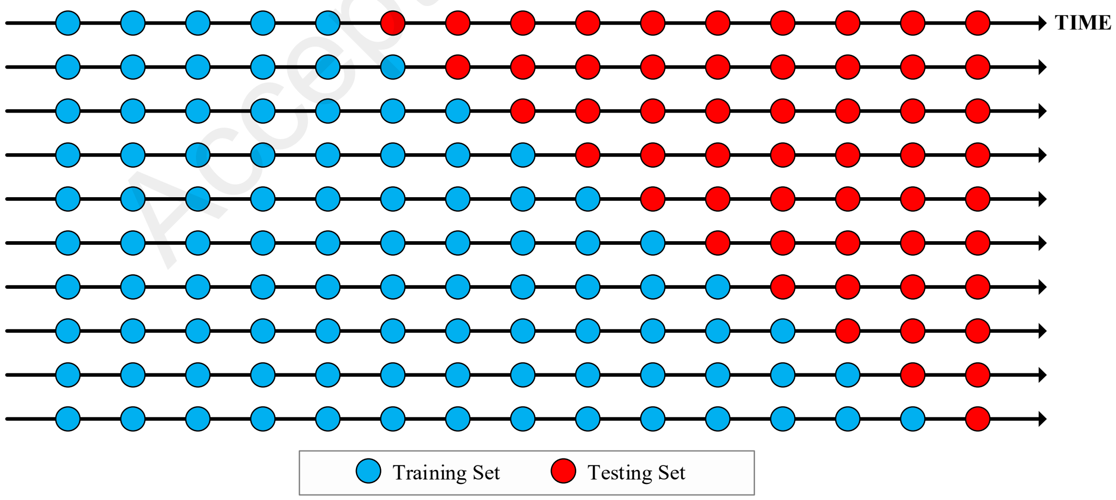


Figure 2: Improvised Rolling-origin Evaluation

3.2 Selected Classifiers for Evaluation

Weka stable version 3.8 is used throughout the experiments. ZeroR, Random Tree, REPtree, Decision Stump (Iba & Langley, 1992), Adaboost (Freund & Schapire, 1996), Bayesnet (Pearl, 1985), Naïve Bayes (John & Langley, 1995), Random Forest (Breiman, 2001), SMO (better known as Supper Vector Machine) (Vapnik & Lerner, 1963) and J48 (better known as C4.8) (Quinlan, 1993) are used to evaluate the performances of each dataset. These 10 classifiers are selected because they are composed of varying types of machine learning techniques, which includes: (i) classifiers for measuring the lowest acceptable performance, e.g.: ZeroR, (ii) base learner (weak learner) which are used as building blocks for ensemble techniques, e.g.: Random Tree and Decision Stump, (iii) ensemble classifiers, e.g.: Adaboost (boosting) and Random Forest (bagging), (iv) probabilistic model, e.g.: Naïve Bayes and Bayesnet, and (v) non-probabilistic model, e.g.: SMO. Additionally, C4.5 decision tree (J48), support vector machine (SMO), Adaboost, and Naïve Bayes are recognized to be few of the most notable and influential data mining algorithms (Wu et al., 2008).

4 Experiments

4.1 Experimental Setup

In this section, experiments with different machine classifiers are conducted on the computing platform with an eight-core 3.64 GHz AMD Ryzen 7 1700 CPU and 64 GB RAM running on Windows 10. To avoid the biasness resulted from the cherry-picked training and testing data distribution elaborated in Section 2.3, the experimental evaluation is performed based on the distribution as described in Section 3.1. The 10 mentioned classifiers from Weka stable version 3.8 mentioned in Section 3.2 are adopted to deliver the empirical results. Attributes and class labels are intentionally not (minimal) modified in this paper in order to conduct a fair comparison between classifiers.

4.2 Datasets Selection

Details and surveys for most of the NIDS datasets were greatly disclosed in (Bhuyan et al., 2014; Mishra et al., 2018; Nicholas et al., 2018). To obtain the empirical results, several experiments have been performed on three publicly available NIDS datasets: GureKDDCup (Perona et al., 2016; Perona et al., 2008), UNSW-NB15 (Moustafa & Slay, 2015), and CIDDS-001 (Ring et al., 2017). The three aforementioned datasets are preferred over the benchmark NIDS datasets as the datasets contain a more recent network traffic and attacks which could better represent the current state of network traffics. The datasets adopted in this paper are summarized in Table 5. It should be noted that GureKDDCup was initially generated and made known to the public in 2008, while the full documentation for the procedure to generate the dataset was released in 2016.

Table 5: Summary of Experimental Datasets

Datasets	**#Used Instances**	**Original Attributes (include Class)**	**#Used Classes**	**Year Released**
Full GureKDDCup	2,759,494	48	36	2008 (2016)
Full UNSW-NB15	2,540,047	49 (raw) 48 (processed)	10	2015
CIDDS-001	18,762,253	12 (raw) 16 (processed)	5	2017

4.3 Datasets Pre-processing

To fairly evaluate the performance of each machine learner, no (minimal) data cleansing and pre-processing are performed against the datasets employed in order to conform to the different requirements of varying classifiers. Each of the description and pre-processing steps for the datasets are thoroughly explained in Sections 4.3.1, 4.3.2, and 4.3.3.

4.3.1 GureKDDCup

GureKDDCup (Perona et al., 2016; Perona et al., 2008) was released in 2008 to complement the drawback of the KDDCup'99 (Stolfo et al., 2000) NIDS datasets. As KDDCup'99 is more than decades old, researchers have deemed the datasets to be obsolete in reflecting the present network traffics (Ahmed et al., 2016; Catania & Garino, 2012). GureKDDCup was generated by utilizing the similar data collecting approach as of KDDCup'99. The dataset contains the network traffic data collected over a period of seven weeks, from Monday to Friday (five days per week). Additionally, the creators of GureKDDCup have included several new attacks that are absent in KDDCup'99. Full version of GureKDDCup dataset are compressed into a 9.3GB file (`gureKddcup.tar.gz`). The file size is tremendously huge due to the added payloads of each of the network traffic. However, only the daily logs found in their respective `gureKddcup-matched.list` are used for the experiment conducted. Each of the daily logs are concatenated and merged into a single file containing a week of network traffic. Table 6 shows the total number of instances available in each week while Table 7 presents the distribution of training and testing data in accordance to the number of weeks. No data cleansing and attribute reduction are necessary to be performed against GureKDDCup dataset.

Table 6: Number of Instances for GureKDDCup in Each Week

Week	**#Instances**
1	177,910
2	188,790
3	288,369
4	113,946
5	604,303
6	951,361
7	434,815
total	2,759,494

Table 7: Train-Test Data Distribution for GureKDDCup

Training Group	**Testing Group**
1	2~7
1~2	3~7
1~3	4~7
1~4	5~7
1~5	6~7
1~6	7

4.3.2 UNSW-NB15

Moustafa and Slay (2015) released the UNSW-NB15 dataset in 2015 to complement for the lack of low footprint attacks in KDDCup'99 (Stolfo et al., 2000). Although the original documentation stated a total number of 2,540,044 instances, an additional instance has been verified and found in all three of the data set files, leading to a total of 2,540,047 instances (Al-Zewairi et al., 2017). As shown in Table 8, some data cleansing is unavoidable, it is necessary to be conducted against the dataset before building the machine classifier. In the raw UNSW-NB15 dataset, some of the values for the *state* and *service* attributes are denoted as '-'. It can be assumed that these values are referring to the missing values since the data owner fails to provide them. As Weka processed missing values as '?' instead of '-', all the raw data containing '-' values are changed to '?'. The *label* attribute consisting of binary class {*normal, attack*} is also discarded, and the *attack_cat* containing 10 different classes is subsequently employed as the class label. Table 9 tabulates the number of instances containing in each file while Table 10 provides the distribution of train-test data.

Table 8: Data Pre-processing for UNSW-NB15

Attribute Name	Original Attribute	Modified Attribute Value
state	-	?
service	-	?
attack_cat	NaN	Normal
	Backdoors	Backdoor
label	(ALL)	Drop
sport / dport	- 0xc0a8	? 49320
	0x000b	11
	0x000c	12
	0x20205321	?
	0xcc09	52233

Table 9: Number of Instances for UNSW-NB15 in Each File

File	#Instances
1	700,001
2	700,001
3	700,001
4	440,044
Total	2,540,047

Table 10: Train-Test Data Distribution for UNSW-NB15

Training Group	Testing Group
1	2~4
1~2	3~4
1~3	4

4.3.3 CIDDS-001

CIDDS-001 (Ring et al., 2017) dataset is publicly available since 2017. The dataset contains a total of 32 million flows, whereby 31 million from the emulated internal environment (OpenStack); and 0.7 million from the external traffic consisting of real traffics from the internet. External traffics have been excluded from the experiment conducted in this paper due to its reduced precision in ground truth, and absence of certain simulated attacks. The full CIDDS-001 OpenStack internal traffics is employed in the experimental procedure. Table 11 shows some of the data cleansing process necessary to be performed on the dataset. The entire *Flows* attribute was removed from the dataset as it contains only a single constant value for all instances (Nicholas et al. 2018). In the case of *Flags*, it has been categorized into five distinct *Flag* in accordance to their value. Three of the decimal values in destination port (*dst_pt*) are converted to '0' as they represent the ICMP error messages instead of the port. The IP addresses are modified in such a way that it won't collides with other IP addresses and matches the dot delimiters of IP addresses. The total number of instances for each week are tabulated in Table 12, while the training and testing data distributions are presented as shown in Table 13. Since the instances in Week 3 and 4 only encompassed of normal traffics, it is reasonable to exclude them from the experiments. Besides, as timestamp attribute (*date_first_seen*) is not supported by Bayesnet, Naïve Bayes, and SMO models, it is removed before building the aforementioned classifiers.

Table 11: Data Pre-processing for OpenStack CIDDS-001

Attribute Name	**Original Attribute Value**	**Modified Attribute Value**
Bytes	(M)	Multiply (Bytes) with 100000
Flows	(ALL)	DROP
Attacktype	---	normal
dst_pt	3.1, 3.2, 3.4	0
Flags	APRSF	Flag_A
		Flag_P
		Flag_R
		Flag_S
		Flag_F
IP Address (Src IP Addr / Dst IP Addr)	DNS	1000.1000.1000.1000
	EXT_SERVER	2000.2000.2000.2000
	(Anonymised IP)	3000.3000.3000.3000

Table 12: Number of Instances for OpenStack CIDDS-001 in Each Week

Week	#Instances
1	8,451,520
2	10,310,733
3	6,349,783
4	6,175,897
total	31,287,933

Table 13: Train-Test Data Distribution for OpenStack CIDDS-001

Training Group	Testing Group
1	2

4.3.4 Discussions on Datasets Class Distribution

As mentioned in Section 4.1, experiments performed in this paper attempts to retain the features and class labels as close as possible to the original datasets to provide a baseline empirical results for future comparison. However, we would like to highlight the concern of imbalance classes in each of the datasets selected and provide some recommendation to be considered for future experiment settings. For instance, the class distributions for each dataset are summarized in Table 14 (GureKDDCup, Table 15 (UNSW-NB15), and Table 16 (CIDDS-001).

GureKDDCup that imitates the generation process of KDDCup'99 attempts to preserve as much class labels that are available in KDDCup'99. Referring to Table 15, a total of 36 classes are found and it can be observed that several classes contain only a small proportion compared to the number of instances in the entire dataset. To improve the performance of the models, the class labels can be commonly redistributed into two classes – benign and anomaly (Kanakarajan & Muniasamy, 2016; Nicholas et al., 2018), or five classes – benign, DoS, user to root (U2L), remote to local (R2L), and Probes (Bouzida & Cuppens, 2006; Nicholas et al., 2018).

On the other hand, UNSW-NB15 contains 10 classes while CIDDS-001 contains only 5 classes. Similarly, the class label in UNSW-NB15 and CIDDS-001 can also be reclassified according to the suggestion made for GureKDDCup. Alternatively, the classification models can also be designed to only detect a specific attack by training the models using only the specific attack sample. To avoid the datasets to immensely skewed towards the normal class, all normal class samples can be excluded for all three of the datasets in the future.

To reiterate, we would like to emphasize that the primary objective of the empirical results presented in this paper are to be used as a baseline results in the future to evaluate the performance for any modification or enhancement in terms of features selection, model enhancement, class redistribution etc. For instance, if the classification results of the models after any feature selection or classes redistribution achieve a superior performance than the empirical results presented in this paper, this would imply that the procedures adopted are able to improve the model. Contrarily, the enhanced models would denote an insignificant improvement in the scenario whereby the model's classification performance is poorer than the baseline results exhibited in this paper. Hence, we aim to minimize the modification performed on the datasets to preserve the originality.

Table 14: Class Distribution for GureKDDCup in Each Week

Attack Type \ Week	1	2	3	4	5	6	7	Total
normal	15	985	203,815	800	449,092	257,406	217,743	1,129,856
anomaly	0	0	1	2	4	2	0	9
back	1	3	198	1798	0	100	148	2,248
dict	0	1	8	41	0	829	0	879
dict_simple	0	0	0	0	1	0	0	1
eject	0	1	0	7	0	2	1	11
eject-fail	0	0	0	0	1	0	0	1
ffb	0	0	1	6	0	2	1	10
ffb_clear	0	0	0	0	0	1	0	1
format	0	0	1	0	2	3	0	6
format_clear	0	0	0	1	0	0	0	1
format-fail	0	0	0	0	0	1	0	1
ftp-write	0	0	1	1	2	4	0	8
guest	0	1	7	17	4	21	0	50
imap	0	0	1	1	1	4	0	7
ipsweep	1	49	1,950	510	7,633	4,851	766	15,760
land	0	1	7	11	0	10	6	35
load_clear	0	0	0	0	1	0	0	1
loadmodule	0	0	1	1	2	3	1	8
multihop	0	0	1	1	4	3	0	9
neptune	177,889	186,706	72,676	98,627	128,516	656,629	205,600	1,526,643
nmap	1	0	49	1,945	0	0	0	1,995
perl_clear	0	0	0	0	0	1	0	1
perlmagic	0	0	0	1	0	2	1	4
phf	0	0	1	0	1	3	0	5
pod	0	0	0	1	0	2	2	5
portsweep	1	14	1,980	361	2,480	2,760	2,377	9,973
rootkit	0	1	6	2	1	17	2	29
satan	1	101	1,986	5,491	6,538	10,406	6,888	31,411
smurf	1	924	5,632	2,509	9,434	17,988	1,178	37,666
spy	0	0	0	1	0	1	0	2
syslog	0	0	0	1	0	2	1	4
teardrop	0	1	18	82	586	298	100	1,085
warez	0	0	0	1	0	0	0	1
warezclient	0	1	29	1,719	0	0	0	1,749
warezmaster	0	1	0	8	0	10	0	19
Total	177,910	188,790	288,369	113,946	604,303	951,361	434,815	2,759,494

Table 15: Class Distribution for UNSW-NB15 in Each File

Attack Type \ File	1	2	3	4	Total
normal	677,786	647,252	542,576	351,150	2,218,764
analysis	526	608	873	670	2,677
backdoor	534	370	759	666	2,329
dos	1,167	4,637	5,642	4,907	16,353
exploits	5,409	11,103	16,574	11,439	44,525
fuzzers	5,051	4,668	9,137	5,390	24,246
generic	7,522	27,883	118,198	61,878	215,481
reconnaissance	1,759	3,116	5,582	3,530	13,987
shellcode	223	324	593	371	1,511
worms	24	40	67	43	174
Total	700,001	700,001	700,001	440,044	2,540,047

Table 16: Class Distribution for CIDDS-001 in Each File

Attack Type \ Week	1	2	3	4	Total
normal	7,010,897	8,515,329	6,349,783 **(not used)**	6,175,897 **(not used)**	28,051,906
dos	1,252,127	1,706,900	-	-	2,959,027
portscan	183,511	82,407	-	-	265,918
pingscan	3,359	2,731	-	-	6,090
bruteforce	1,626	3,366	-	-	4,992
Total	8,451,520	10,310,733	6,349,783	6,175,897	31,287,933

4.4 Evaluation Metrics

Empirical results are reported based on the classification accuracy, which is a standard performance evaluation metric used in the data mining community. As the NIDS classes are often imbalanced and are skewed towards the normal class, classification accuracy metric might not be sufficient to measure the effectiveness of a machine model (Tavallaee, 2011). Thus, detection rate and false positive rate are also included in the scope of experimental discussion to complement the drawback of classification accuracy. Additionally, both of the metrics are deemed as one of the most commonly used metrics in this domain to evaluate the performance of a NIDS model based on the survey conducted by Tavallaee (2011). The conventional confusion matrix of performance measurement is shown in Figure 3.

Classification accuracy, which is also known as the percentage of successful prediction are commonly adopted for gauging the overall performance of a classifiers. It can be formed from the confusion matrix as shown in Figure 3 as follows:

$$Accuracy = \frac{TP + TN}{P + N} \tag{1}$$

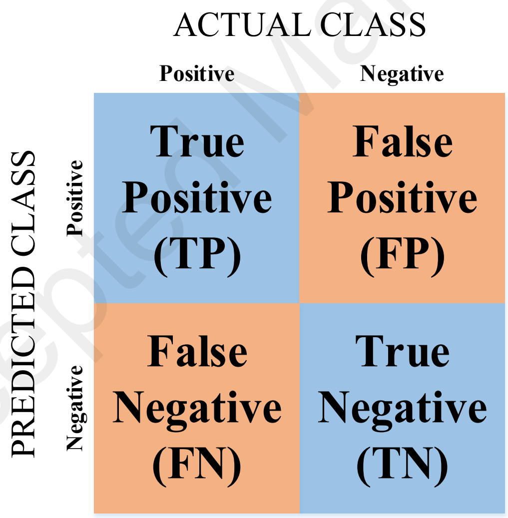


Figure 3: Performance Measurement Confusion Matrix

Detection rate is the proportion of positive case (an intrusion or attack) correctly identified as an attack. In some literatures, detection rate can also be referred as true positive rate, recall, or sensitivity. The detection rate formula is expressed in Equation 2 as follows:

$$Detection\ Rate = \frac{TP}{TP + FN} \quad (2)$$

False positive rate measures the proportion of negative case (a benign or normal traffic) incorrectly identified as an attack. The term false alarm rate or false acceptance rate can also be denoted to signify equivalent meaning as false positive rate. Equation 3 presents the formulation of false positive rate:

$$False\ Positive\ Rate = \frac{FP}{FP + TN} \quad (3)$$

4.5 Experimental Results

4.5.1 Classification Accuracy, Detection Rate, False Positive Rate

Empirical results obtained by the 10 machine models on the NIDS datasets are tabulated in Supplementary Table 1 (classification accuracy), Supplementary Table 2 (detection rate), and Supplementary Table 3 (false positive rate). As shown in Supplementary Table 1 – Supplementary Table 3, the performance of a model varies when a distinct set of train-test is supplied to the classifiers.

In terms of classification accuracy, the best results for GureKDDCup obtained are as follows: 88.5261% – Naïve Bayes [Train:1; Test:2~7], 87.1651% – Random Tree [Train:1~2; Test:3~7], 87.2992% – SMO [Train:1~3; Test:4~7], and 88.1692% / 88.8412% / 99.9526% – J48 [Train: 1~4 / 1~5 / 1~6; Test: 5~7 / 6~7 / 7]. Among the 10 classifiers, Random Tree score the best average accuracy of 80.1090% in GureKDDCup. On the other hand, J48 models outperform the other classifiers in UNSW-NB15 while

Adaboost in CIDDS-001 datasets. To be specific, the J48 algorithm achieve an average accuracy of 96.6325% in UNSW-NB15 while Adaboost attained 96.8684% in CIDDS-001.

Detection rate obtained by the 10 classifiers are obviously very encouraging, as most of them managed to attain a relatively decent performance. However, by observing the detection rates attained in Supplementary Table 2, we noticed that the empirical values delivered by each of the classifiers are similar to the classification accuracies in Supplementary Table 1. After analyzing the number of normal instances in each of the datasets, we discovered that the distributions are greatly skewed towards the normal class across the three datasets, as shown in Table 17. Although there is only 40.9443% of normal instances in GureKDDCup, the number of classes are particularly large when comparing against UNSW-NB15 and CIDDS-001. As most of the NIDS datasets attempt to mirror the scenarios in real world application, they often include unseen attacks (classes) in the test set. Hence, this causes some of the attacks to be undetected by the models and subsequently leading to an unknown detection rate.

Table 17: Number of Normal Instances in Entire GureKDDCup, UNSW-NB15, and CIDDS-001

Datasets	#Number of Normal Instances	#Number of Total Instances	Ratio (%) of Normal : Total Instances	#Used Classes
GureKDDCup	1,129,856	2,759,494	40.9443	36
UNSW-NB15	2,218,764	2,540,047	87.3513	10
CIDDS-001	15,526,226	18,762,253	82.7523	5

False positive rate empirical results are presented in Supplementary Table 3. In many cases, a high detection rate might indirectly result in a high false positive rate. However, Bayesnet is able to achieve a low false positive rate in GureKDDCup and UNSW-NB15 datasets while maintaining satisfactory detection rate (Supplementary

Table 2). In particular, an average detection rate of 98.3662% with 0.0271% false positive rate in GureKDDCup [Train:1~6; Test: 7] and 94.3472% of detection rate with 0.0994 false positive rate in UNSW-NB15 [Train: 1~3; Test: 4]. Conversely, all of the models provide a high false positive rate in CIDDS-001 except Adaboost that is capable to achieve a false positive rate of 4.5325% along with an eminent detection rate of 96.8684%.

Supplementary Table 4 summarizes the effectiveness of the machine classifiers on the three NIDS datasets. It can be observed that there is no universal classifier that excels across all problem domains, in which the highest accuracy is achieved by Bayesnet in GureKDDCup, J48 in UNSW-NB15, and Adaboost in CIDDS-001. Although Naïve Bayes (1.0413%) attain a lower false positive rate in comparison towards J48 (2.6059%), Naïve Bayes (79.5543%) detection rate is widely inferior when comparing against J48 (96.6325%) in UNSW-NB15.

From Supplementary Table 1 – Supplementary Table 3, it is worth to note that the classification accuracy for SMO is denoted as 'n/a' (not available) when using GureKDDCup week 1~6 as training data and week 7 as testing data. This is due to the failed attempt of building the classification model even after running for 604,800 seconds (~7 days) as tabulated in Supplementary Table 5. For comparison purposes in the future, the unknown accuracy, detection rate, false positive rate, and evaluation time for the aforementioned SMO model can follows the preceding results (GureKDDCup week 1~5 training data against week 6~7 testing data).

4.5.2 Build Time and Evaluation Time

Apart from the classification accuracy, computation time is also one of the vital aspects when considering the application in a real-world scenario. Computation time to build the models is tabulated in Supplementary Table 5 while the time taken to evaluate each of the models is summarized in Supplementary Table 6. By analyzing the results, we can observe that the computation time is acceptable for most of the classifiers except REPtree, Random Forest and SMO in some cases. To be specific, the time taken required to build the classifiers exceeds 86,400 seconds (1 day) when it is train with week 1~6 of GureKDDCup. For instance, REPtree took 87,704.84 seconds, Random Forest required 97,996.45 second, and SMO needed more than 604,800 seconds (~7 days) to completely build the models. Besides, we also noticed a fluctuation in terms of memory resources (RAM), whereby some of the classifiers consumed the entire 64GB RAM during the experimental procedure.

The huge consumption of computation time is speculated to be precipitated by the massive amount of discrete (nominal) IP feature, as shown in Table 18. It is worth to point out that most classifiers might treat each the features (e.g., source IP and destination IP) as a separate feature or entity during the building and classification process. Thus, the collision of similar IP address in both source and destination is also calculated in the number of unique source and number of destination IP. For example, if the source IP of "192.168.1.1" is also seen in the destination IP features, both the number of unique source and destination IP are increased by one. However, the assumption made on the effects of the high dimensionality IP features in affecting the computation time required further investigation and justification to corroborate the claim. In addition, this scenario may also possibly cause by the algorithm behind the machine classifiers. Hence, examination can also be carried out directly on each of the classifiers to investigate the reasons for the classifiers to exhaust tremendous resources. As the primary objective of this paper is to deliver a baseline empirical results, in-depth discussions for each of the classifiers will not be deliberated.

Table 18: Number of Unique IP Address in Entire GureKDDCup, UNSW-NB15, and CIDDS-001

Datasets	#Number of Unique Source IP	#Number of Unique Destination IP	#Total Number of Unique IP Address
GureKDDCup	8,483	21,018	29,501
UNSW-NB15	43	47	90
CIDDS-001	38	790	828

4.5.3 Experimental Results of Cross-Validation

To substantiate the claim made in Section 2.3 regarding the inappropriateness of adopting cross-validation in the domain of NIDS, we have also performed the experiments employing the identical experimental setup, datasets, and evaluation metrics. Supplementary Table 7 presents the classification accuracy, detection rate, and false positive rate while Supplementary Table 8 tabulates the time taken evaluate the machine classifiers.

As accentuate by Al Tobi and Duncan (2019), cross-validation will yield an over-optimistic performance results. This claim is further corroborated by the experimental results shown in Supplementary Table 7. It can be observed that most of the classifiers achieve a remarkable boost in terms of performance. For instance, the classification accuracy attains by J48 decision tree achieved approximately 99% on all the three datasets when cross-validation is utilized. These over-optimistic empirical results might not be able to reflect the genuine performance of the classifiers.

Referring to Supplementary Table 8, we noticed the time taken required to evaluate the models is longer as compared to Supplementary Table 5 and Supplementary

Table 6. This observation is common as it is necessary to repeat the training and testing procedure for 10 times when 10 cross-validation sampling method is adopted. Theoretically, the time taken for building and evaluating the models are multiplied by 10. Akin to the observation in Section 4.5.2, a vast amount of computation time is required to evaluate the classifiers. It is also worth to mentioned there are more machine classifiers that are denoted with 'n/a' in Supplementary Table 7 and Supplementary Table 8 as it required an extensive span of time to complete the experiment. To be specific, random tree, REPtree, random forest, and SMO fails to be evaluated in GureKDDCup, while random forest and SMO fails to be evaluated in CIDDS-001. Although SMO have been successfully evaluated in UNSW-NB15, the time taken is considerable huge and it is not impractical in a real-world scenario. For instance, 2,159,708 seconds is needed to complete the 10 cycles of training and testing process. Similar to the observation in Section 4.5.2, we surmise that the vast amount of computation time might be also caused by the immense number of IP address features.

4.5.4 Performance Compatibility with 5 Benchmark Feature Selection Techniques in UNSW-NB15

Though the main contribution of this paper is to provide a benchmark study of different classifiers, but it is also important to observe their compatibility with other feature selection techniques. A wise use of feature selection technique can definitely bring many benefits, i.e., removing the agitated noises and improving the classifiers' performance (Guyon & Elisseeff, 2003).

Thus, in this section, we would like to perform the same set of testing but with the additional usage of feature selection techniques, including Association Rule Mining (Moustafa & Slay, 2017), Information Gain (Janarthanan & Zargari, 2017), Principal

Component Analysis (Moustafa et al., 2018), Gain Ratio Filter (Anwer et al., 2018), and XGBoost (Kasongo & Sun, 2020) in UNSW-NB15 dataset. To provide a fair comparison among the feature selection techniques, we adopted the best-selected features in each of the studies and recomputes the results based on the identical experimental settings as described in section 4.1 and section 4.3.2.

UNSW-NB15 is selected to be rigorously experimented in this section because it contains the largest number of attributes (i.e., 49 features) among the three datasets. As the same experimental settings should be employed to fairly compares the classification performance, we utilized the *attack_cat* along with its 10 distinct classes as described in Section 4.3.2 to conduct the experiment. The best-selected features in each of the works are tabulated in Table 19. It should be noted that only 17 attributes in Anwer, Farouk and Abdel-Hamid (2018) and 18 attributes in Kasongo and Sun (2020) work are utilized because the attribute of "*rate*" is not available in the full version of UNSW-NB15.

Classification accuracy for each of the distinct selected features is tabulated in Supplementary Table 9. To demonstrate the usability of the baseline results delivered in Section 4.5.1, a scatter plot based on the baseline accuracy and 5 feature selection approach for each of the machine classifiers is presented in Supplementary Figure 1. In most cases, it can be seen that ZeroR, Decision Stump and Adaboost is less likely to be affected by the features selected. Generally, the performance of Random Tree, REPtree, Bayesnet, Naïve Bayes, Random Forest, SMO and J48 combining with feature selection techniques surpassed the baseline results. In specific, it can be observed that the performance of the mentioned 7 classifiers has been improved when adopting the 18 features selected by XGBoost. However, we also noticed a significant degradation in terms of performance when using the 8 features chosen by Principal Component Analysis for the 7 classifiers.

Based on the empirical results, a suitable feature selection technique can undeniably boost the performance of the classifiers by eliminating noises. On the contrary, unsuitable feature selection techniques can also be dangerous as it may discard valuable attributes which can be employed to build a better classifier.

Table 19: Features Selected for the 5 Distinct Feature Selection Techniques

Authors	**Original Dataset**	**Moustafa and Slay (2017)**	**Janarthanan and Zargari (2017)**	**Moustafa, Creech and Slay (2018)**	**Answer, Farouk, and Abdel-Hamid (2018)**	**Kasongo and Sun (2020)**
Feature Selection Tech-nique	-	Association Rule Mining	CfsSubsetEval (attribute evaluator) + GreedyStepwise method + InfoGainAttributeEval + Ranker Method (Weka)	Principal Component Analysis	Gain Ratio Filter	XGBoosts
Number of Feature Selected	49 (including 2 labels)	11	5	8	18 (use 17)	19 (use 18)
No.	**Features**					
1	srcip					
2	sport					
3	dstip					
4	dsport					
5	proto					✓
6	state	✓			✓	✓
7	dur				✓	
8	sbytes		✓		✓	✓
9	dbytes				✓	✓
10	sttl	✓	✓		✓	✓
11	dttl	✓			✓	
12	sloss					✓
13	dloss					✓
14	service		✓	✓		✓
15	Sload					
16	Dload				✓	
17	Spkts					
18	Dpkts				✓	
19	swin	✓				
20	dwin	✓		✓		
21	stcpb					
22	dtcpb					
23	smeansz		✓	✓		✓
24	dmeansz				✓	✓

25	trans_depth					
26	res_bdy_len				✓	
27	Sjit					
28	Djit	✓				
29	Stime					
30	Ltime					
31	Sintpkt					
32	Dintpkt				✓	
33	tcprtt			✓	✓	✓
34	synack	✓			✓	✓
35	ackdat				✓	
36	is_sm_ips_ports				✓	
37	ct_state_ttl	✓			✓	✓
38	ct_flw_http_mthd					
39	is_ftp_login					
40	ct_ftp_cmd					
41	ct_srv_src					✓
42	ct_srv_dst	✓				✓
43	ct_dst_ltm			✓		
44	ct_src_ ltm	✓				
45	ct_src_dport_ltm		✓	✓		✓
46	ct_dst_sport_ltm	✓		✓	✓	✓
47	ct_dst_src_ltm			✓		✓
48	attack_cat					
49	Label					

5 Conclusion and Future Works

In this paper, a standard resampling method – rolling-origin is adopted to allocate the train-test distribution of the full version of three NIDS dataset (GureKDDCup, UNSW-NB15 and CIDDS-001). Subsequently, 10 notable machine classifiers (ZeroR, Random Tree, REPtree, Decision Stump, Adaboost, Bayesnet, Naïve Bayes, Random Forest, SMO, and J48) are employed to evaluate on the selected three NIDS datasets. The results are presented with five evaluation metrics including classification accuracy, detection rate, false positive rate, building time, and evaluation time. Empirical results in this paper will be served as a baseline comparison for other studies performed on these datasets.

Due to privacy concerns and the massive efforts required to label NIDS dataset, Ring et al. (2019) highlighted that it is impossible to create a perfect NIDS dataset. A perfect NIDS dataset should keep up to date with recent attacks, correctly labelled, publicly available, contain real network traffic with all class of attacks, and captured over a long period of time (Ring et al., 2019). Assume that real network traffics with various attacks have been captured without privacy and computational resources constrain, the time taken necessary to accurately label the data would indefinitely delayed the publishing process. As new attacks are observed in every single day, these delayed would cause the dataset to be slightly outdated when it is published. For the details of traits and properties of a good benchmark NIDS dataset, we refer to the studies conducted by Małowidzki et al. (2015) and Ring et al. (2019). Although it is not possible to build a perfect NIDS dataset, Ring et al. (2019) pointed out that a perfect dataset is not necessary for most application, but instead a good dataset that fulfil a certain property is sufficient. For example, to evaluate a new reconnaissance technique, the NIDS dataset are not expected to contain all kinds of attacks.

Among the three NIDS datasets that have been rigorously experimented in this paper, UNSW-NB15 is appropriate for attack detection in low footprint scenario, CIDDS-001 is suitable for detecting reconnaissance techniques, while GureKDDCup contain most of the features akin to the previous benchmark KDDCup'99 and can be used in place of the previous benchmark dataset.

For future works, rolling-origin resampling methods can be utilized in other NIDS datasets which do not comprise of a pre-defined train-test distribution. To gain a better insight of detection rate, we suggest to reduce the number of classes in GureKDDCup, whereby the 36 classes can be reduced to five classes (normal, probe, denial-of-service, user-to-root, and remote-to-local) or with only binary classes (normal, anomaly). Since the results in this paper should serve as a baseline, experimental procedure adopting the smaller number of classes are excluded. Future experiments employing the smaller number of classes could utilize the baseline results provided in this paper for comparison purpose, which allows for the assessment of a model's performance gain or loss. On the other hand, experiments that attempt to reduce the enormous number of discrete values in IP features should also be considered to substantiate the speculation made on the impacts of high dimensionality IP attributes on the computation time of the models in Section 4.5.2.

Acknowledgements

This research work was supported by a Fundamental Research Grant Schemes (FRGS) under the Ministry of Education and Multimedia University, Malaysia (Project ID: MMUE/160029), and Korea Foundation of Advanced Studies (ISEF).

References

Abas, E. A. E. R., Abdelkader, H., & Keshk, A. (2015). Artificial immune system based intrusion detection. *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, 542–546.

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, *60*, 19–31. https://doi.org/10.1016/j.jnca.2015.11.016

Al-Riyami, S., Coenen, F., & Lisitsa, A. (2018). A Re-evaluation of Intrusion Detection Accuracy: Alternative Evaluation Strategy. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2195–2197.

Al-Zewairi, M., Almajali, S., & Awajan, A. (2017). Experimental evaluation of a multi-layer feed-forward artificial neural network classifier for network intrusion detection system. *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, 167–172.

Al Tobi, A. M., & Duncan, I. (2019). Improving Intrusion Detection Model Prediction by Threshold Adaptation. *Information*, *10*(5), 159.

Althubiti, S. A., Jones, E. M., & Roy, K. (2018). LSTM for Anomaly-Based Network Intrusion Detection. *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, 1–3.

Anwer, H. M., Farouk, M., & Abdel-Hamid, A. (2018). A framework for efficient network anomaly intrusion detection with features selection. *2018 9th International Conference on Information and Communication Systems (ICICS)*, 157–162.

Baig, M. M., Awais, M. M., & El-Alfy, E.-S. M. (2017). A multiclass cascade of artificial neural network for network intrusion detection. *Journal of Intelligent & Fuzzy Systems*, *32*(4), 2875–2883.

Bamakan, S. M. H., Wang, H., & Shi, Y. (2017). Ramp loss K-Support Vector Classification-Regression; a robust and sparse multi-class approach to the intrusion detection problem. *Knowledge-Based Systems*, *126*, 113–126. https://doi.org/10.1016/j.knosys.2017.03.012

Belouch, M., El Hadaj, S., & Idhammad, M. (2017). A two-stage classifier approach using reptree algorithm for network intrusion detection. *International Journal of Advanced Computer Science and Applications (Ijacsa)*, *8*(6), 389–394.

Benmessahel, I., Xie, K., & Chellal, M. (2017). A new evolutionary neural networks based on intrusion detection systems using multiverse optimization. *Applied Intelligence*, *48*(8), 2315–2327.

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192–213. https://doi.org/10.1016/j.ins.2011.12.028

Bermingham, M. L., Pong-Wong, R., Spiliopoulou, A., Hayward, C., Rudan, I., Campbell, H., Wright, A. F., Wilson, J. F., Agakov, F., & Navarro, P. (2015). Application of high-dimensional feature selection: evaluation for genomic prediction in man. *Scientific Reports*, *5*, 10312.

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network Anomaly Detection : Methods , Systems and Tools. *IEEE Communications Surveys & Tutorials*, *16*(1), 303–336. https://doi.org/10.1109/SURV.2013.052213.00046

Bouzida, Y., & Cuppens, F. (2006). Neural networks vs. decision trees for intrusion detection. *Communications, 2006. ICC '06. IEEE International Conference On*, 2394–2400. https://doi.org/10.1016/j.jmaa.2011.01.050

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Catania, C. A., & Garino, C. G. (2012). Automatic network intrusion detection: Current techniques and open issues. *Computers and Electrical Engineering*, *38*(5), 1062–1072. https://doi.org/10.1016/j.compeleceng.2012.05.013

Chawla, N. V, Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chen, Z.-H., & Tsai, C.-W. (2018). An Effective Metaheuristic Algorithm for Intrusion Detection System. *2018 IEEE International Conference on Smart Internet of Things (SmartIoT)*, 154–159.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, *1*(3), 131–156.

Divekar, A., Parekh, M., Savla, V., Mishra, R., & Shirole, M. (2018). Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives. *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, 1–8.

Elhag, S., Fernández, A., Altalhi, A., Alshomrani, S., & Herrera, F. (2017). A multi-objective evolutionary fuzzy system to obtain a broad and accurate set of solutions in intrusion detection systems. *Soft Computing*, *23*(4), 1321–1336. https://doi.org/10.1007/s00500-017-2856-4

Elmasry, W., Akbulut, A., & Zaim, A. H. (2019). Empirical study on multiclass classification-based network intrusion detection. *Computational Intelligence*, *35*(4), 1–36.

Freund, Y., & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. *International Conference on Machine Learning*, *96*, 148–156. https://doi.org/10.1.1.133.1040

Gharaee, H., & Hosseinvand, H. (2016). A new feature selection IDS based on genetic algorithm and SVM. *2016 8th International Symposium on Telecommunications (IST)*, 139–144.

Guha, S., Yau, S. S., & Buduru, A. B. (2016). Attack detection in cloud infrastructures using artificial neural network with genetic feature selection. *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech*, 414–419.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*(Mar), 1157–1182.

Hyndsight, R. J. (2016). *Cross-validation for time series*. https://robjhyndman.com/hyndsight/tscv/

Iba, W., & Langley, P. (1992). Induction of one-level decision trees. In *Machine Learning Proceedings 1992* (pp. 233–240). Elsevier.

Idhammad, M., Afdel, K., & Belouch, M. (2017). Dos detection method based on artificial neural networks. *International Journal of Advanced Computer Science and Applications*, *8*(4), 465–471.

Idhammad, M., Afdel, K., & Belouch, M. (2018a). Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest. *Security and Communication Networks*.

Idhammad, M., Afdel, K., & Belouch, M. (2018b). Distributed intrusion detection system for cloud environments based on data mining techniques. *Procedia Computer Science*, *127*, 35–41.

Idhammad, M., Afdel, K., & Belouch, M. (2018c). Semi-supervised machine learning approach for DDoS detection. *Applied Intelligence*, *48*(10), 3193–3208.

Ikram, S. T., & Cherukuri, A. K. (2016). Improving accuracy of intrusion detection model using pca and optimized svm. *Journal of Computing and Information Technology*, *24*(2), 133–148.

Jabbar, M. A., Aluvalu, R., & Reddy, S. (2017). Cluster based ensemble classification for intrusion detection system. *Proceedings of the 9th International Conference on Machine Learning and Computing*, 253–257.

Janarthanan, T., & Zargari, S. (2017). Feature selection in UNSW-NB15 and KDDCUP'99 datasets. *2017 IEEE 26th International Symposium on Industrial*

Electronics (ISIE), 1881–1886.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 338–345.

Kanakarajan, N. K., & Muniasamy, K. (2016). Improving the accuracy of intrusion detection using GAR-Forest with feature selection. *Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015*, 539–547.

Kasongo, S. M., & Sun, Y. (2020). Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset. *Journal of Big Data*, *7*(1), 1–20.

Kayacık, H. G., & Zincir-Heywood, N. (2005). Analysis of three intrusion detection system benchmark datasets using machine learning algorithms. *International Conference on Intelligence and Security Informatics*, 362–367.

Kohavi, R. (1995). A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, 1137–1143.

Mahoney, M. V, & Chan, P. K. (2003). An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. In G. Vigna, C. Kruegel, & E. Jonsson (Eds.), *International Workshop on Recent Advances in Intrusion Detection* (pp. 220–237). Springer Berlin Heidelberg.

Małowidzki, M., Berezi, P., & Mazur, M. (2015). *Network Intrusion Detection : Half a Kingdom for a Good Dataset* (pp. 1–6).

Masduki, B. W., & Ramli, K. (2016). Improving intrusion detection system detection accuracy and reducing learning time by combining selected features selection and parameters optimization. *2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 397–402.

McHugh, J. (2000). Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, *3*(4), 262–294. https://doi.org/10.1145/382912.382923

Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2018). A Detailed Investigation and Analysis of using Machine Learning Techniques for Intrusion Detection. *IEEE Communications Surveys & Tutorials*, *21*(1), 686–728. https://doi.org/10.1109/COMST.2018.2847722

Moustafa, N., Creech, G., & Slay, J. (2018). Anomaly detection system using beta mixture models and outlier detection. In *Progress in Computing, Analytics and Networking* (pp. 125–135). Springer.

Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal*, *25*(1–3), 18–31. https://doi.org/10.1080/19393555.2015.1125974

Moustafa, N., & Slay, J. (2017). A hybrid feature selection for network intrusion detection systems: Central points. *ArXiv Preprint ArXiv:1707.05505*.

Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems. *2015 Military Communications and Information Systems Conference (MilCIS)*, 1–6. https://doi.org/10.1109/MilCIS.2015.7348942

Ng, A. (2015). *Andrew Ng: Why 'Deep Learning' Is a Mandate for Humans, Not Just Machines*. https://www.wired.com/brandlab/2015/05/andrew-ng-deep-learning-mandate-humans-not-just-machines/

Nicholas, L., Ooi, S. Y., Pang, Y. H., Hwang, S. O., & Tan, S.-Y. (2018). Study of long short-term memory in flow-based network intrusion detection system. *Journal of Intelligent & Fuzzy Systems*, *35*(6), 5947–5957. https://doi.org/10.3233/JIFS-169836

Patil, D. R., & Pattewar, T. M. (2014). A Comparative Performance Evaluation of Machine Learning-Based NIDS on Benchmark Datasets. *International Journal of Research in Advent Technology*, *2*(2), 101–106.

Pearl, J. (1985). Bayesian networks: A model of self-activated memory for evidential reasoning. *Proceedings of the 7th Conference of the Cognitive Science Society, 1985*, 329–334.

Perona, Inigo, Arbelaitz, O., Gurrutxaga, I., Martin, J. I., Muguerza, J., & Perez, J. M. (2016). *Generation of the database gurekddcup*.

Perona, Iñigo, Gurrutxaga, I., Arbelaitz, O., Martín, J. I., Muguerza, J., & Mª, J. (2008). Service-independent payload analysis to improve intrusion detection in network traffic. *In Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, 171–178. https://dl.acm.org/citation.cfm?id=2449315

Primartha, R., & Tama, B. A. (2017). Anomaly detection using random forest: A performance revisited. *2017 International Conference on Data and Software Engineering (ICoDSE)*, 1–6.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.

Ring, M., Landes, D., & Hotho, A. (2018). Detection of slow port scans in flow-based network traffic. *PloS One*, *13*(9), 1–18.

Ring, M., Wunderlich, S., Grüdl, D., Landes, D., & Hotho, A. (2017). Flow-based benchmark data sets for intrusion detection. *Proceedings of the 16th European Conference on Cyber Warfare and Security*, 361–369. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85028023805&partnerID=40&md5=1e95f767994dde4a33199aa24418b078

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A Survey of Network-based Intrusion Detection Data Sets. *Computers & Security*, *86*, 147–167.

Sahu, S. K., & Jena, S. K. (2014). A Study of K-Means and C-Means Clustering Algorithms for Intrusion Detection Product Development. *International Journal of Innovation, Management and Technology*, *5*(3), 207–213. https://doi.org/10.7763/IJIMT.2014.V5.515

Sahu, S. K., & Jena, S. K. (2016). A Multiclass SVM Classification Approach for Intrusion Detection. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 175–181. https://doi.org/10.1007/978-3-319-28034-9

Sainis, N., Srivastava, D., & Singh, R. (2018). Feature Classification and Outlier Detection to Increased Accuracy in Intrusion Detection System. *International Journal of Applied Engineering Research*, *13*(10), 7249–7255.

Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection.

Computers and Security, *31*(3), 357–374. https://doi.org/10.1016/j.cose.2011.12.012

Singh, S., & Silakari, S. (2009). A survey of cyber attack detection systems. *International Journal of Computer Science and Network Security*, *9*(5), 1–10.

Sommer, R., & Paxson, V. (2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *2010 IEEE Symposium on Security and Privacy*, 305–316. https://doi.org/10.1109/SP.2010.25

Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A., & Chan, P. K. (2000). Cost-based modeling for fraud and intrusion detection: results from the JAM project. *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, *2*, 130–144. https://doi.org/10.1109/DISCEX.2000.821515

Tama, B. A., & Rhee, K.-H. (2017). Attack classification analysis of IoT network via deep learning approach. *Research Briefs on Information & Communication Technology Evolution*, *3*, 1–9.

Tashman, L. J. (2000). Out-of Sample Tests of Forecasting Accuracy: A Tutorial and Review. *International Journal of Forecasting*, *16*(4), 437–450.

Tavallaee, M. (2011). *An adaptive hybrid intrusion detection system*. University of New Brunswick, Faculty of Computer Science.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, *Cisda*, 1–6. https://doi.org/10.1109/CISDA.2009.5356528

Timčenko, V., & Gajin, S. (2017). Ensemble classifiers for supervised anomaly based network intrusion detection. *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 13–19.

Vapnik, V., & Lerner, A. (1963). Pattern recognition using generalized portrait method. *Automation and Remote Control*, *24*, 774–780.

Verma, A., & Ranga, V. (2018a). On evaluation of network intrusion detection systems: Statistical analysis of CIDDS-001 dataset using machine learning techniques. *Pertanika Journal of Science & Technology*, *26*(3), 1307–1332.

Verma, A., & Ranga, V. (2018b). Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning. *Procedia Computer Science*, *125*, 709–716.

Wang, H., Gu, J., & Wang, S. (2017). An effective intrusion detection framework based on SVM with feature augmentation. *Knowledge-Based Systems*, *136*, 130–139. https://doi.org/10.1016/j.knosys.2017.09.014

Wu, X., Kumar, V., Ross, Q. J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z. H., Steinbach, M., Hand, D. J., & Steinberg, D. (2008). Top 10 algorithms in data mining. In *Knowledge and Information Systems* (Vol. 14, Issue 1). https://doi.org/10.1007/s10115-007-0114-2

Zhou, Y., Han, M., Liu, L., He, J. S., & Wang, Y. (2018). Deep learning approach for cyberattack detection. *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 262–267.

Zhu, Y., Liang, J., Chen, J., & Ming, Z. (2017). An improved NSGA-III algorithm for feature selection used in intrusion detection. *Knowledge-Based Systems*, *116*, 74–85. https://doi.org/10.1016/j.knosys.2016.10.030

Appendices

Supplementary Table 1: Classification Accuracy of 10 Machine Classifiers on GureKDDCup, UNSW-NB15, and CIDDS-001

Dataset		Classification Accuracy (%) of 10 Machine Classifiers for Original Data									
Train	*Test*	ZeroR	Random Tree	REPtree	Decision Stump	Adaboost	Bayesnet	Naïve Bayes	Random Forest	SMO	J48
GureKDDCup											
1	2~7	36.8753	36.8746	36.8753	36.8753	36.8753	75.3595	**88.5261**	38.0152	78.5929	38.0293
1~2	3~7	31.9819	**87.1651**	31.9844	31.9831	31.9831	73.4939	62.8376	31.9958	28.8684	33.1808
1~3	4~7	32.9109	85.0984	62.2119	62.1153	62.1153	77.2154	81.3405	86.1029	**87.2992**	82.8830
1~4	5~7	29.8400	87.8231	60.9671	60.7162	60.7162	86.6704	54.5618	87.9254	83.7842	**88.1692**
1~5	6~7	33.5774	84.3049	82.4255	81.6088	81.6088	84.4639	79.2033	84.1741	83.4074	**88.8412**
1~6	7	47.2845	99.3880	97.6507	95.7251	95.7251	98.3662	87.4997	99.5161	n/a	**99.9526**
Average		35.4117	80.1090	62.0192	61.5040	61.5040	**82.5949**	75.6615	71.2883	72.3904	71.8427
UNSW-NB15											
1	2~4	83.7467	91.7973	90.3676	92.9147	92.9147	92.8976	72.6009	94.7403	23.5704	**96.5079**
1~2	3~4	78.3939	86.3023	90.2963	91.5745	91.5745	78.6976	77.0988	96.5630	90.9196	**96.6146**
1~3	4	79.7988	79.1657	82.0959	91.6120	91.6120	94.3472	88.9631	96.6649	94.0967	**96.7751**
Average		80.6465	85.7551	87.5866	92.0337	92.0337	88.6475	79.5543	95.9894	69.5289	**96.6325**
CIDDS-001											
1	2	82.5870	78.9227	71.5996	85.5956	**96.8684**	85.4093	56.3338	84.6270	75.1466	94.4539

Not available (n/a)

Supplementary Table 2: Detection Rate of 10 Machine Classifiers on GureKDDCup, UNSW-NB15, and CIDDS-001

Dataset		Detection Rate (%) of 10 Machine Classifiers for Original Data									
Train	*Test*	ZeroR	Random Tree	REPtree	Decision Stump	Adaboost	Bayesnet	Naïve Bayes	Random Forest	SMO	J48
GureKDDCup											
1	2~7	36.8753	36.8746	36.8753	36.8753	36.8753	75.3595	**88.5261**	38.0152	78.5929	38.0293
1~2	3~7	31.9819	**87.1651**	31.9845	31.9831	31.9831	73.4939	62.8376	31.9958	28.8684	33.1808
1~3	4~7	32.9109	85.0984	62.2119	62.1153	62.1153	77.2154	81.3405	86.1029	**87.2992**	82.8830
1~4	5~7	29.8400	87.8231	60.9671	60.7162	60.7162	86.6704	54.5618	87.9254	83.7842	**88.1692**
1~5	6~7	33.5774	84.3049	82.4255	81.6088	81.6088	84.4639	79.2033	84.1741	83.4074	**88.8412**
1~6	7	47.2845	99.3880	97.6507	95.7251	95.7251	98.3662	87.4997	99.5161	n/a	**99.9526**
Average		35.4117	80.1090	62.0192	61.5040	61.5040	**82.5949**	75.6615	71.2883	72.3904	71.8427
UNSW-NB15											
1	2~4	83.7467	91.7973	90.3676	92.9147	92.9147	92.8976	72.6009	94.7403	23.5704	**96.5079**
1~2	3~4	78.3939	86.3023	90.2963	91.5745	91.5745	78.6976	77.0988	96.5630	90.9196	**96.6146**
1~3	4	79.7988	79.1657	82.0959	91.6120	91.6120	94.3472	88.9631	96.6649	94.0967	**96.7751**
Average		80.6465	85.7551	87.5866	92.0337	92.0337	88.6475	79.5543	95.9894	69.5289	**96.6325**
CIDDS-001											
1	2	82.5870	78.9227	71.5996	85.5956	**96.8684**	85.4093	56.3338	84.6270	75.1466	94.4539

Not available (n/a)

Supplementary Table 3: False Positive Rate of 10 Machine Classifiers on GureKDDCup, UNSW-NB15, and CIDDS-001

Dataset		False Positive Rate (%) of 10 Machine Classifiers for Original Data									
Train	*Test*	ZeroR	Random Tree	REPtree	Decision Stump	Adaboost	Bayesnet	Naïve Bayes	Random Forest	SMO	J48
GureKDDCup											
1	2~7	36.8753	36.8753	36.8753	36.8753	36.8753	**0.2528**	7.9014	36.2095	8.3542	36.2325
1~2	3~7	31.9819	6.0243	27.8922	31.9519	31.9519	**0.2862**	15.0886	31.7733	11.3712	27.2800
1~3	4~7	32.9109	6.4898	13.3309	18.5845	18.5845	**0.2734**	5.9948	6.4003	0.6449	1.8845
1~4	5~7	29.8400	4.9389	16.0890	16.7079	16.7079	**0.0532**	16.9361	1.5526	0.3385	0.3975
1~5	6~7	33.5774	0.7966	8.3137	9.7326	9.7326	**0.0843**	3.7122	7.8700	0.4087	0.9563
1~6	7	47.2845	0.3538	0.3464	3.9880	3.9880	**0.0271**	0.2218	0.3603	n/a	0.0374
Average		35.4117	9.2465	17.1413	19.6400	19.6400	**0.1628**	8.3092	14.0277	4.2235	11.1314
UNSW-NB15											
1	2~4	83.7467	13.0765	10.3127	0.9937	0.9937	**0.1095**	0.3064	12.6090	1.5891	3.3015
1~2	3~4	78.3939	25.2890	2.2885	**1.6602**	**1.6602**	3.0296	2.5036	2.9460	21.6017	1.9251
1~3	4	79.7988	4.4179	2.5511	1.6487	1.6487	**0.0994**	0.3140	2.0899	13.1853	2.5911
Average		80.6465	14.2611	5.0508	1.4342	1.4342	1.0795	**1.0413**	5.8816	12.1254	2.6059
CIDDS-001											
1	2	82.5870	65.1739	71.0452	6.7301	**4.5325**	68.1673	59.0684	69.4820	69.2468	25.9203

Not available (n/a)

Supplementary Table 4: Best Results for Classification Accuracy, Detection Rate, and False Positive Rate of 10 Machine Classifiers on GureKDDCup, UNSW-NB15, and CIDDS-001

Dataset		Accuracy		Detection Rate		False Positive Rate	
Train	*Test*	Classifier	(%)	Classifier	(%)	Classifier	(%)
GureKDDCup							
1	2~7	Naïve Bayes	88.5261	Naïve Bayes	88.5261	Bayesnet	0.2528
1~2	3~7	Random Tree	87.1651	Random Tree	87.1651	Bayesnet	0.2862
1~3	4~7	SMO	87.2992	SMO	87.2992	Bayesnet	0.2734
1~4	5~7	J48	88.1692	J48	88.1692	Bayesnet	0.0532
1~5	6~7	J48	88.8412	J48	88.8412	Bayesnet	0.0843
1~6	7	J48	99.9526	J48	99.9526	Bayesnet	0.0271
Average		**Bayesnet**	**82.5949**	**Bayesnet**	**82.5949**	**Bayesnet**	**0.1628**
UNSW-NB15							
1	2~4	J48	96.5079	J48	96.5079	Bayesnet	0.1095
1~2	3~4	J48	96.6146	J48	96.6146	Decision Stump / Adaboost	1.6602
1~3	4	J48	96.7751	J48	96.7751	Bayesnet	0.0994
Average		**J48**	**96.6325**	**J48**	**96.6325**	**Naïve Bayes**	**1.0413**
CIDDS-001							
1	2	**Adaboost**	**96.8684**	**Adaboost**	**96.8684**	**Adaboost**	**4.5325**

Supplementary Table 5: Build Time of 10 Machine Classifiers on GureKDDCup, UNSW-NB15, and CIDDS-001

Dataset *Train*	*Test*	Model Building Time (seconds) of 10 Machine Classifiers for Original Data: ZeroR	Random Tree	REPtree	Decision Stump	Adaboost	Bayesnet	Naïve Bayes	Random Forest	SMO	J48
GureKDDCup											
1	2~7	0.06	1.05	232.97	9.03	140.91	28.23	1.37	36.39	225.27	10.56
1~2	3~7	0.03	1.99	131.31	12.33	131.69	33.25	2.88	154.37	1102.89	99.42
1~3	4~7	0.05	107.74	2473.48	29.55	127.56	65.71	4.99	3849.94	4356.92	242.68
1~4	5~7	0.05	7.49	4189.84	24.25	125.07	92.85	5.76	7719.14	7738.95	242.65
1~5	6~7	0.10	30.38	**43050.39**	52.67	130.13	148.13	30.13	**38090.68**	**210451**	495.51
1~6	7	0.19	3154.34	**87704.84**	65.94	19.76	318.14	220.77	**97996.45**	***604800***	1053.40
Average		0.08	550.50	**22963.81**	32.30	112.52	114.39	44.32	**24641.16**	**138112.51**	357.37
UNSW-NB15											
1	2~4	0.42	12.45	24.00	17.42	131.75	84.11	11.49	540.50	1949.02	112.70
1~2	3~4	0.17	29.82	52.22	40.16	202.85	159.82	21.60	1557.24	**26547.31**	372.31
1~3	4	0.32	17.92	92.78	53.47	322.51	241.47	34.99	2225.02	**153171**	1091.03
Average		0.30	20.06	56.33	37.02	219.04	161.80	22.69	1440.92	**60555.78**	525.35
CIDDS-001											
1	2	5.80	41.18	811.30	68.73	486.31	139.47	14.20	**16062.12**	**24979.81**	522.52

Time required > 10800 seconds (3 hours) are bold

Supplementary Table 6: Evaluation Time of 10 Machine Classifiers on GureKDDCup, UNSW-NB15, and CIDDS-001

Dataset		Model Evaluation Time (seconds) of 10 Machine Classifiers for Original Data									
Train	*Test*	ZeroR	Random Tree	REPtree	Decision Stump	Adaboost	Bayesnet	Naïve Bayes	Random Forest	SMO	J48
GureKDDCup											
1	2~7	204.20	215.37	383.04	287.12	5.73	555.52	1303.68	1505.20	1516.46	203.53
1~2	3~7	203.47	198.97	770.50	301.58	3.55	611.53	1159.68	**14817.11**	3188.98	175.10
1~3	4~7	135.69	546.64	1341.99	180.24	2.56	449.83	972.80	**47115.73**	7965.54	136.13
1~4	5~7	134.58	128.72	1549.12	181.00	1.79	444.09	5438.14	**46678.89**	8683.84	331.35
1~5	6~7	97.78	78.70	91.80	130.54	1.67	266.58	**74750.26**	**19576.22**	**13660.68**	81.11
1~6	7	5.10	2377.31	13.13	16.37	219.58	55.88	**29836.95**	**20955.79**	n/a	6.03
Average		130.14	590.95	691.60	182.81	39.15	397.24	**18910.25**	**25108.16**	7003.10	155.54
UNSW-NB15											
1	2~4	5.69	5.37	6.28	22.14	33.62	68.19	2587.53	272.22	143.09	50.14
1~2	3~4	2.86	3.09	3.23	3.14	18.22	40.95	1604.80	244.04	78.32	22.67
1~3	4	1.08	1.22	1.25	1.18	9.09	17.11	592.69	339.76	30.97	5.86
Average		3.21	3.23	3.59	8.82	20.31	42.08	1595.01	285.34	84.13	26.22
CIDDS-001											
1	2	17.19	54.57	474.11	129.14	97.10	45.72	108.09	**50907.07**	1021.85	55.55

Not available (n/a)

Time required > 10800 seconds (3 hours) are bold

Supplementary Table 7: Classification Accuracy, Detection Rate and False Positive Rate of 10 Machine Classifiers on GureKDDCup, UNSW-NB15, and CIDDS-001 using 10-Fold Cross-Validation

Dataset *10-fold Cross Validation*	Classification Accuracy (%) of 10 Machine Classifiers									
	ZeroR	Random Tree	REPtree	Decision Stump	Adaboost	Bayesnet	Naïve Bayes	Random Forest	SMO	J48
GureKDDCup	55.3233	n/a	n/a	95.1351	95.1351	99.6401	85.1627	n/a	n/a	**99.9795**
UNSW-NB15	87.3513	98.0091	98.3296	94.1896	94.1896	96.3009	93.0825	98.3747	97.4715	**98.5983**
CIDDS-001	82.7525	99.9578	99.9587	85.0414	92.2821	99.8454	48.2678	n/a	n/a	**99.9686**
	Detection Rate (%) of 10 Machine Classifiers									
GureKDDCup	0.5532	n/a	n/a	0.9514	0.9514	0.9964	0.8516	n/a	n/a	**0.9998**
UNSW-NB15	0.8735	0.9801	0.9833	0.9419	0.9419	0.9630	0.9308	0.9837	0.9747	**0.9860**
CIDDS-001	0.8275	0.9996	0.9996	0.8504	0.9228	0.9985	0.4827	n/a	n/a	**0.9997**
	False Positive Rate (%) of 10 Machine Classifiers									
GureKDDCup	0.5532	n/a	n/a	0.0501	0.0501	**0.0001**	0.0013	n/a	n/a	**0.0001**
UNSW-NB15	0.8735	0.0156	0.0121	0.0084	0.0084	0.0004	**0.0010**	0.0105	0.0606	0.0102
CIDDS-001	0.8275	**0.0010**	0.0011	0.0958	0.0825	0.0020	0.0147	n/a	n/a	0.0011

Not available (n/a)

Supplementary Table 8: Evaluation Time of 10 Machine Classifiers on GureKDDCup, UNSW-NB15, and CIDDS-001 using 10-Fold Cross-Validation

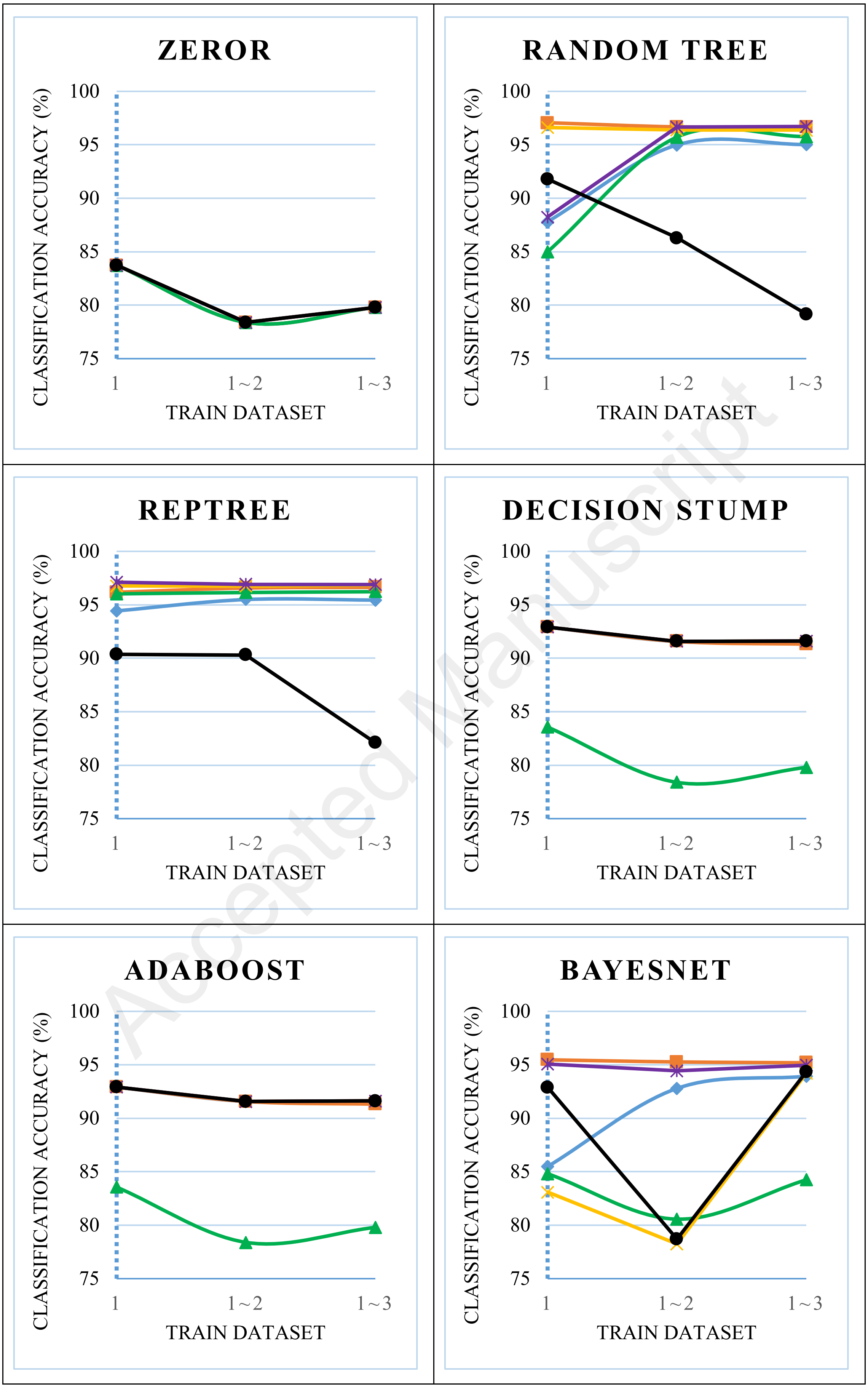
Dataset *10-fold Cross Validation*	Model Evaluation Time (seconds) of 10 Machine Classifiers									
	ZeroR	Random Tree	REPtree	Decision Stump	Adaboost	Bayesnet	Naïve Bayes	Random Forest	SMO	J48
GureKDDCup	23.18	n/a	n/a	508.90	1772.47	2260.53	11613.92	n/a	n/a	**18035.24**
UNSW-NB15	19.59	238.07	882.91	526.40	2327.19	2268.52	589.18	**29096.73**	**2159708.79**	**23004.56**
CIDDS-001	177.67	**69816.38**	**3083528.47**	1243.59	7936.36	3589.40	581.44	n/a	n/a	**36550.10**

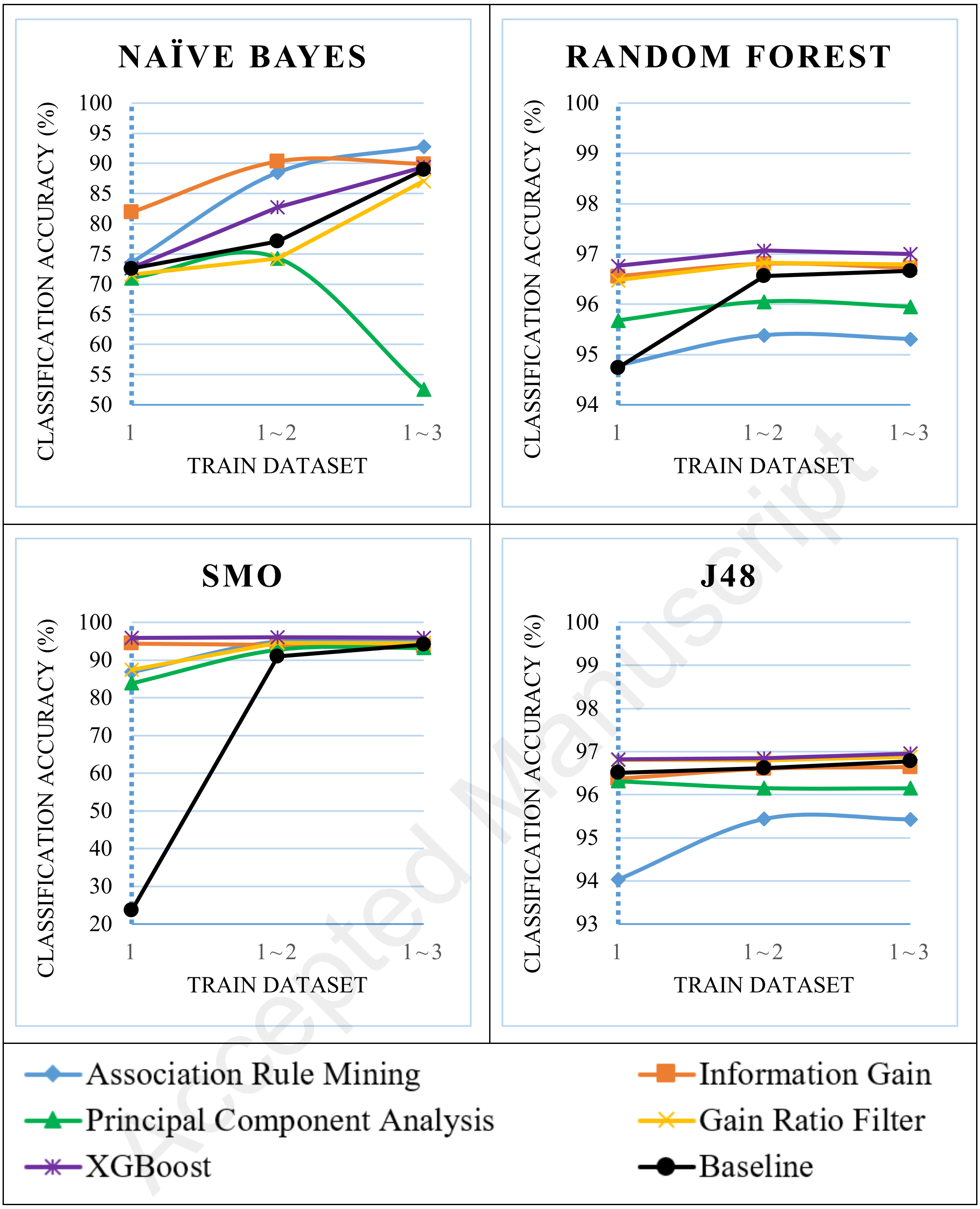
Not available (n/a) Time required > 10800 seconds (3 hours) are bold

Supplementary Table 9: Classification Accuracy of 10 Machine Classifiers on UNSW-NB15 using 5 Feature Selection Techniques

Dataset *Train*	 *Test*	Classification Accuracy (%) of 10 Machine Classifiers using various Feature Selection Technique									
		ZeroR	Random Tree	REPtree	Decision Stump	Adaboost	Bayesnet	Naïve Bayes	Random Forest	SMO	J48
Moustafa and Slay (2017) – Association Rule Mining											
1	2~4	83.7467	87.8075	94.4311	92.9147	92.9147	85.4812	73.5953	**94.7880**	86.7895	94.0351
1~2	3~4	78.3939	94.9908	**95.4943**	91.5745	91.5745	92.7926	88.4947	95.3841	94.9561	95.4376
1~3	4	79.7988	95.0232	**95.4407**	91.6120	91.6120	93.9317	92.7928	95.3116	94.9030	95.4300
Average		80.6465	92.6072	95.1220	92.0337	92.0337	90.7352	84.9609	**95.1612**	92.2162	94.9676
Janarthanan and Zargari (2017) – Attribute Evaluator + Greedystepwise + Information Gain Attribute Evalutor + Ranker											
1	2~4	83.7467	**97.0687**	96.1727	92.9147	92.9147	95.4683	81.9252	96.5609	94.3444	96.3725
1~2	3~4	78.3939	96.6811	96.5717	91.5745	91.5745	95.2693	90.3268	**96.8058**	94.0175	96.6035
1~3	4	79.7988	96.6910	96.6331	91.3213	91.3213	95.1891	89.9392	**96.7242**	93.6104	96.6419
Average		80.6465	**96.8136**	96.4592	91.9368	91.9368	95.3089	87.3971	96.6970	93.9908	96.5393

Dataset Train	Test	ZeroR	Random Tree	REPtree	Decision Stump	Adaboost	Bayesnet	Naïve Bayes	Random Forest	SMO	J48
		Classification Accuracy (%) of 10 Machine Classifiers using various Feature Selection Technique									
Moustafa, Creech and Slay (2018) – Principal Component Analysis											
1	2~4	83.7467	84.9974	96.0241	83.5619	83.5619	84.8155	71.0558	95.6810	83.8246	**96.3137**
1~2	3~4	78.3939	95.7234	96.1527	78.3939	78.3939	80.5646	74.3325	96.0559	92.7109	**96.1545**
1~3	4	79.7988	95.7686	**96.2249**	79.7988	79.7988	84.2475	52.5572	95.9513	93.2230	96.1479
Average		80.6465	92.1631	96.1339	80.5849	80.5849	83.2092	65.9818	95.8961	89.9195	**96.2054**
Anwer, Farouk and Abdel-Hamid (2018) – Gain Ratio Filter											
1	2~4	83.7467	96.6198	96.7628	92.9147	92.9147	83.0921	71.5820	96.4831	87.4090	**96.8069**
1~2	3~4	78.3939	96.4007	96.7513	91.5745	91.5745	78.2478	74.2159	**96.8257**	94.5487	96.7983
1~3	4	79.7988	96.3847	96.8387	91.6120	91.6120	94.2013	87.1208	96.7892	94.5153	**96.8824**
Average		80.6465	96.4684	96.7843	92.0337	92.0337	85.1804	77.6396	96.6993	92.1577	**96.8292**
Kasongo and Sun (2020) – XGBoost											
1	2~4	83.7467	88.2560	**97.1124**	92.9147	92.9147	95.0714	72.6963	96.7667	95.8702	96.8220
1~2	3~4	78.3939	96.6838	96.9249	91.5745	91.5745	94.4414	82.7535	**97.0694**	96.0434	96.8465
1~3	4	79.7988	96.7153	96.8946	91.6120	91.6120	94.9632	89.4142	**97.0033**	95.9247	96.9533
Average		80.6465	93.8850	**96.9773**	92.0337	92.0337	94.8253	81.6213	96.9465	95.9461	96.8739


ZEROR
RANDOM TREE
REPTREE
DECISION STUMP
ADABOOST
BAYESNET
CLASSIFICATION ACCURACY (%)
100
95
90
85
80
75
1
1~2
1~3
TRAIN DATASET



Supplementary Figure 1: Classification Accuracy Comparison of 10 Machine Classifiers on UNSW-NB15 by utilizing the Baseline Results (Supplementary Table 1) against five distinct feature selection schemes (Supplementary Table 9)